REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	November 30, 1995	Final: 15 Mar 92 - 1 Oct 95

4. TITLE AND SUBTITLE
Distributed Real-Time Situation Assessment Using the DRESUN System

5. FUNDING NUMBERS
G
N00014-92-J-1450

6. AUTHOR(S)
Victor R. Lesser, PI

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Computer Science Department
Lederle Graduate Research Center, Box 34610
University of Massachusetts
Amherst, MA 01003-4610

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Office of Naval Research
800 N. Quincy Street
Arlington, VA 22217

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The goal of this research has been the development of a generic architecture for large-scale distributed real-time situation assessment systems (DSA). The major areas of effort in this research have been in:

- creating a highly flexible testbed, with agents that can support a very wide range of interaction protocols;
- extending the capabilities of the agents to model the beliefs of other agents, to facilitate communication and use of incomplete information and information at multiple levels of abstraction;
- developing appropriate coordination algorithms/protocols;
- integrating real-time deadlines into the control framework.

We have also been pursuing several different techniques for formally characterizing/analyzing important aspects of DSA systems and applications, and developing techniques for on-line learning of coordination strategies.

14. SUBJECT TERMS	15. NUMBER OF PAGES
REAL-TIME DISTRIBUTED SITUATION ASSESSMENT; MULTIAGENT SYSTEMS; COORDINATION; INTERPRETATION; DISTRIBUTED AI	128
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT

Distributed Real-Time Situation Assessment Using the DRESUN System

Final Technical Report
Period: 3/15/92-10/1/95

Victor R. Lesser, ***Principal Investigator***

Distributed Artificial Intelligence Laboratory
Computer Science Department, Box 34610
Lederle Graduate Research Center
University of Massachusetts
Amherst, MA 01003-4610
(413) 545-1322

Sponsored by Office of Naval Research
Grant Number: N00014-92-J-1450
Contract Period: 3/15/92–10/1/95
Contract Dollars: $732,796

CONTENTS

1 NUMERICAL PRODUCTIVITY MEASURES

Refereed papers submitted but not yet published: 1

Refereed papers published: 32

Unrefereed reports and articles: 18

Books or parts thereof submitted but not yet published: 2

Books or parts thereof published: 2

Thesis: 1

Patents filed but not yet granted: 0

Patents granted: 0

Invited presentations: 4

Contributed presentations: 17

Honors received: 21

Prizes or awards received: 0

Promotions obtained: 0

Graduate Students supported: 9

Post-docs supported: 2

Minorities supported: 0

2 Introduction

Three years ago, we initiated an ambitious, long-term effort to develop a generic architecture for large-scale distributed real-time situation assessment systems (DSA). Significant progress has been made toward this goal through the development of the major components that are necessary for such an architecture and through the investigation of approaches for formally analyzing DSA systems and applications.

Research into a generic agent architecture for real-time DSA holds much promise because it would greatly facilitate the development of a variety of military and commercial applications of DSA and related systems. Examples of such applications include: distributed sensor networks, distributed network diagnosis, distributed information retrieval and digital libraries, distributed perceptual processing for cooperating robots/autonomous vehicles, and computer-supported cooperative work on situation analysis tasks.[1] DSA will also be an integral component of many distributed planning and scheduling systems.

We see these kinds of applications emerging much sooner than we had anticipated three years ago as a result of the quickening development of the NII. Recently, there has been significant effort directed toward building the hardware/software infrastructure that makes these kinds of applications possible because it allows large numbers of intelligent agents to interact over both local and long-distance networks seamlessly. An example of such emerging software infrastructure is the Telescript architecture from General Magic and high-level protocols such as KQML.

Despite these infrastructure advances, there remain many software-related questions that must be answered in order to develop large-scale distributed situation assessment systems. We see the key issues as how to organize both local agent and network-wide problem solving so that the agents can cooperate effectively to produce answers of appropriate quality within fixed deadlines, using limited communication bandwidth, and have their performance degrade gracefully as sensors, communication links, and processors fail.

To address these issues we have been investigating the use of the DRESUN system as the basis for a generic architecture for distributed real-time situation assessment. This architecture was chosen because our earlier research suggested that sophisticated, self-aware agents were required to support the flexible and dynamic problem-solving strategies that are necessary in complex DSA tasks. The major areas of effort in this research to date have been in:

- creating a highly flexible testbed, with agents that can support very wide range of interaction protocols;
- extending the capabilities of the agents to model the beliefs of other to facilitate the communication and use of incomplete information information at multiple levels of abstraction;
- developing appropriate coordination algorithms/protocols;
- integrating real-time deadlines into the control framework.

We have also been pursuing several different techniques for formally characterizing/ analyzing important aspects of DSA systems and applications. We have made significant progress on these areas.

---

[1] It is this last application for which we see a near-term use for our previous research results, and for which we are proposing an optional technology demonstration project.

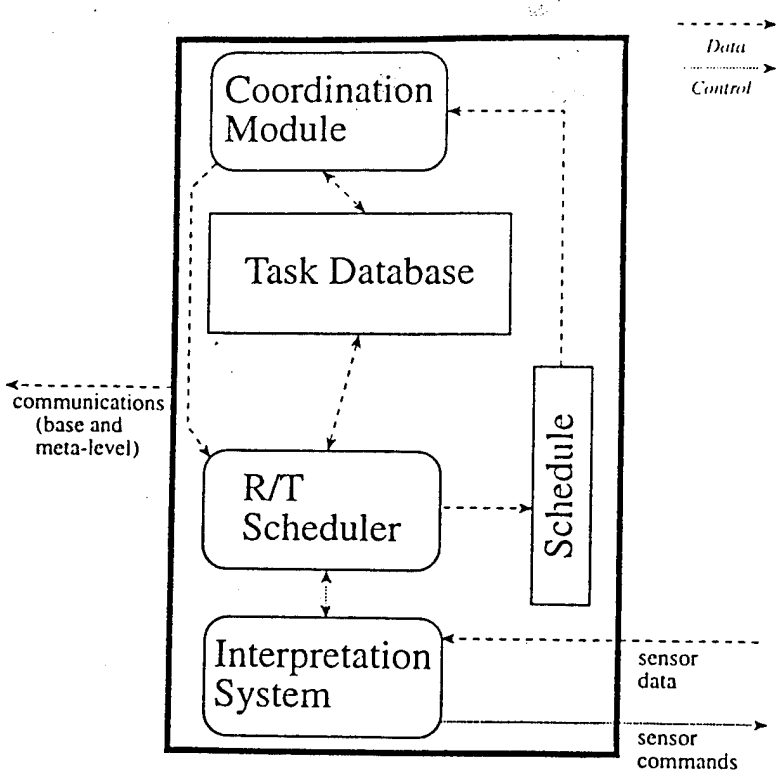


Figure 1: Our model of agents for distributed real-time situation assessment.

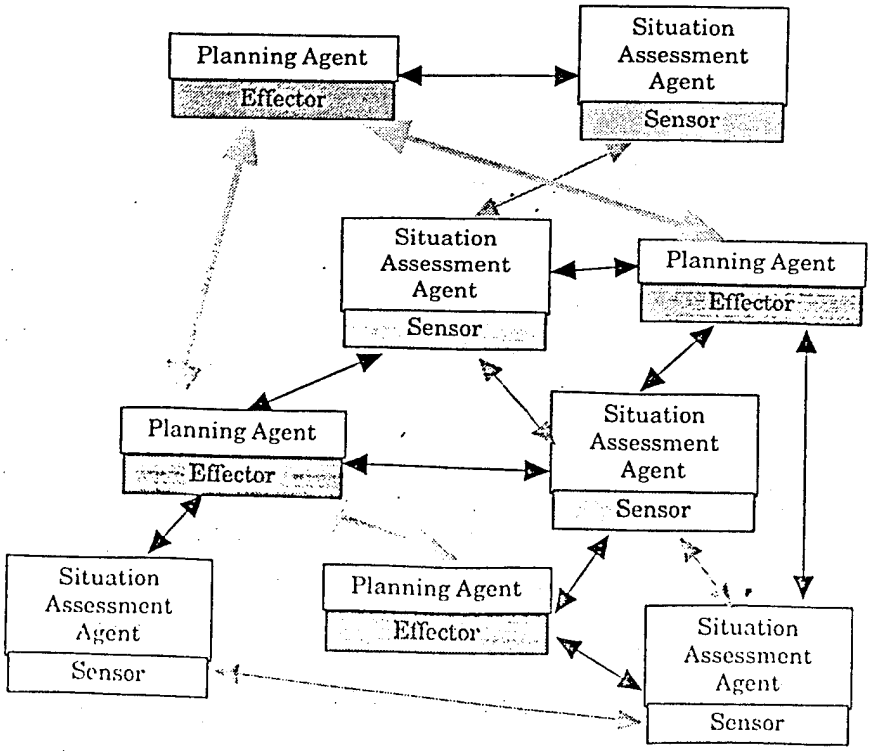


Figure 2: Embedding a DSA system in a complex distributed problem-solving system.

Our recent work has emphasized the development of an agent model that can support sophisticated coordination strategies in the context of real-time constraints. This model is shown in Figure 1. The *task database* maintains a representation of the local agent's tasks and the tasks of other agents, with information about the relations among the tasks. Local task information is provided by the *interpretation system* module, which is also reponsible for low-level control decisions, or when there are no relevant task relationships (subproblem interactions) or real-time constraints. When there are relevant task relationships and real-time constraints, the *real-time scheduler* uses the task information and interacts with the interpretation module to select appropriate tasks/actions. We are currently working on the integration of the interpretation module with the other modules and expect the further development of this model will be an important focus of continued research.

In addition to the progress that we have made developing the key elements of a generic DSA architecture, many of the issues that we are exploring are of general interest for research on cooperative distributed problem-solving systems. For example, a key issue we have had to address is the *bounded rationality* of the agents, which results from limited computational, communications, and sensing resources. This is important because it affects both the satisficing control strategies and the agent coordination strategies by introducing interpretation and control uncertainty since agents are inherently limited to incomplete and possibly inconsistent views of the situation and the problem-solving activities of other agents. Another key issue our research is addressing is how to deal with interacting subproblems in sophisticated knowledge-based systems. The formal analysis work that we have done is beginning to give us the tools to understand how complex problem-solving architectures interact with the characteristics of particular domains.

The next section gives an overview of the key research issues that must be addressed in complex DSA systems and outlines our basic research approach. This is followed by a section that examines the key areas in this research that are of general interest—i.e., that are applicable beyond DSA. A summary of the progress we have made so far is given in Section 5.

3 Overview of the Key Issues in this Research

The key issue in building complex DSA systems is how to organize local problem solving in each agent (node) so that the agents can cooperate to achieve globally coherent behavior — i.e., so that they can work together effectively to get a consistent and accurate assessment of the entire situation. Coherent distributed problem solving means, in part, that the system must have the ability to produce answers of appropriate quality, in a timely manner, using limited communication bandwidth. It is also important that performance degrades gracefully as processors, sensors, and communication links fail. Coherence is a difficult issue because each agent has a limited viewpoint on the developing solution, the available data, and the problem-solving activities of the other agents. This can result in agents doing redundant work, failing to produce results that are necessary for other agents to proceed, communicating unnecessarily, etc.

The main source of these difficulties is the issue of subproblem interactions—i.e., the agents' local solutions are often not independent. This occurs whenever data (evidence) for an interpretation hypothesis is spread among multiple agents or when agent interest areas overlap as a result of overlapping sensor coverage. When agent subproblems interact, the construction of a global solution may not be straightforward—e.g., local agent interpretations may be inconsistent because they are based on different incomplete subsets of the data. In such situations, significant agent interactions may be required if the globally most likely interpretations are to be determined. However, both local problem solving and

inter-agent communications must be carefully controlled to avoid excessive communication costs, delays, and redundant work.

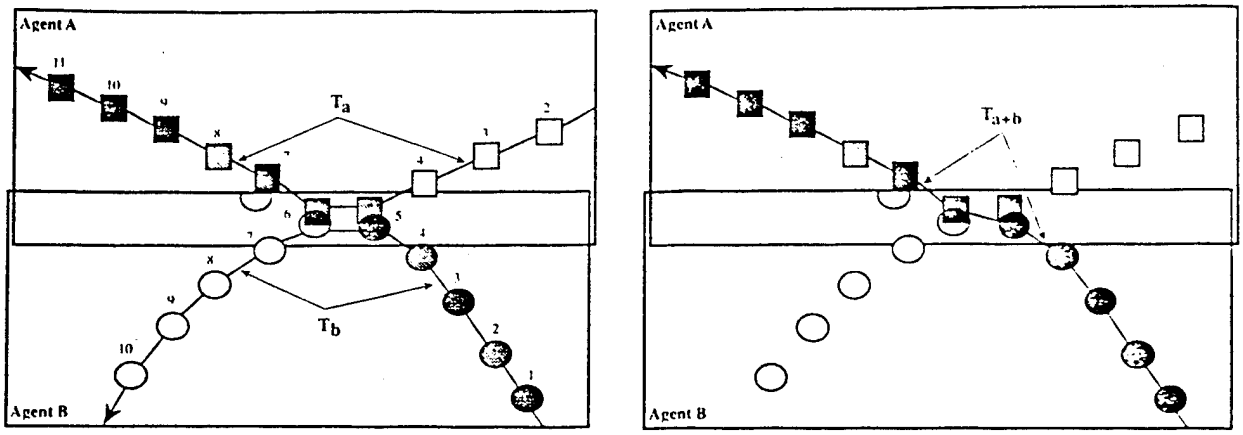


Figure 3: An example of inconsistent local interpretations.

The application is vehicle monitoring. Agent A and Agent B receive data only from their own individual sensors, whose coverage regions overlap. Agent A's data is represented by squares and agent B's by circles, with positions as indicated at the times denoted by the associated numbers. The grey density of the data points corresponds to the relative "quality" of the data—i.e., the a priori likelihood that the data would have resulted from a real vehicle. "Empty" points denote data whose existence has been assumed by the agents. Based on its own data, each agent would form the local interpretations shown: agent A would hypothesize vehicle track T_a and agent would hypothesize vehicle track T_b. T_a covers agent A's data from times 2 through 11, and T_b covers agent B's data from times 1 through 10. These tracks are inconsistent since they imply that either a single vehicle is in different places at the same time or else two vehicles are in the same place at the same time. This inconsistency cannot be immediately resolved because neither T_a nor T_b is significantly more likely than the other (each includes some good quality data and some poor quality data). The preferred global interpretation—given a complete view of the data from both agent A and agent B—is T_{a+b} because it covers more high quality data than either of the local tracks (the remaining uninterpreted data is due to ghosting phenomena and may or may not be explicitly interpreted depending on the termination criteria of the system). T_{a+b} covers agent B's data from times 1 through 6 and agent A's data from times 5 through 11 (it covers both agents' consistent data at times 5 and 6).

The scenario in Figure 3 is an example of a situation in which local solutions are inconsistent, and extended agent interactions are necessary to resolve the inconsistency. In the example, their own local data will cause agent A and agent B to form track hypotheses (T_a and T_b, respectively) that are inconsistent with each other. Because the tracks extend through an area of overlapping interest, the agents recognize that they can communciate to try to verify the global consistency of their local interpretations. Simply exchanging the partial solutions—i.e., the track hypotheses without their supporting evidential structures—is sufficient to allow the inconsistency to be detected, but this level of information is not sufficient to allow the inconsistency to be resolved. Resolving the inconsistency in favor of the most likely global interpretation requires an understanding of the quality of the supporting data for the different portions of each track in order to be able to identify the most likely overall interpretation of the data.

Obviously, one way to insure that agents have the information necessary to resolve global inconsistencies would be to always communicate the complete evidential information associated with the solution hypotheses. However, because interpretation hypotheses are complex structures that may be interrelated with numerous other hypotheses, communication and processing limitations typically make it impractical to fully communicate. [2]. Furthermore, complete communication is usually not necessary—e.g., in this example each agent does not have to know the other's actual data. What is needed is a system with the flexibility to request or respond with information at different levels of detail—based on the dynamic problem-solving requirements—as part of an extended process of resolving inconsistency. This requires the ability to integrate incomplete information, represent the resulting uncertainty, and use this uncertainty to drive further actions.

Our approach to distributed situation assessment is based on the *functionally accurate, cooperative* (FA/C) paradigm for distributed problem solving. In the FA/C paradigm, an agent's local problem solving is organized so that partial and tentative results can be produced despite the agent having incomplete and uncertain information. Inter-agent subproblem interactions are viewed as constraints on the agents' local solutions. When these partial results are exchanged among agents working on interdependent subproblems, these constraints can be exploited to partially resolve the inconsistencies and uncertainties that occur in local problem solving due to the lack of accurate, complete, and up-to-date information. Resolution can take the form of producing more complete partial results, resolving solution uncertainty due to competing, alternative partial solutions, detecting inconsistencies in previously generated results (either locally generated or received from other agents), and speeding up local problem solving because the space of possible solutions that needs to be examined is constrained. The advantage of the FA/C approach is that it reduces that amount of information that must be transferred among agents and reduces delays that would result from the need for synchronization of agent problem solving.

As a result of earlier research on distributed problem solving, we came to understand that a sophisticated agent architecture is necessary to develop distributed real-time situation assessment systems. In this research, we have been investigating the use of the DRESUN system as the basis of a generic architecture for distributed real-time situation assessment. DRESUN agents maintain explicit representations of the reasons why their interpretations are uncertain as well as explicit representations of their goals and the state of problem solving. Because of this, the agents are self-aware and can implement highly context-specific strategies. This makes it possible to support the complex agent interaction protocols that are necessary for multi-sensor fusion and distributed differential diagnosis strategies. It also allows agents to reason about sensor and node errors, to dynamically adjust their problem-solving strategies, and to base termination of network problem solving on realistic measures of the global consistency of local solutions. Our initial experimentation has supported our belief the DRESUN architecture provides the basis for the kind of flexible and reactive approach to the communication and use of external evidence that is necessary to solve the sort of example given above.

One reason why a sophisticated agent architecture is needed is the inherent complexity of situation assessment tasks such as sensor interpretation. In order to understand this point, it is useful to understand the distinction Clancey draws between *classification problem solving* and *constructive problem solving*, and to understand the differences between interpretation problems and the ``diagnosis problems'' that are typically studied in research

[2] The example shown here is simplified to allow us to focus on our main points. It shows only a small fraction of the data that would need to be processed by most real-world DSA systems.

on abductive inference and probabilistic network inference. In classification problem solving, the solution is selected from among a *pre-enumerated* set of all the possible solutions. In constructive problem solving, the set of possible solutions is determined as part of the problem-solving process. Diagnosis problems can be approached using classification techniques because these problems are propositional (they have a fixed set of causes and possible findings, with fixed relations among them). By contrast, while interpretation problems have a fixed set of top-level cause *types* and a fixed set of data *types,* they can have an indeterminate number of instances of any of the types. In particular, there can be an indeterminate number of instances of any top-level cause (e.g., vehicles in a vehicle monitoring system). This leads to the problem of *correlation ambiguity* (it is ambiguous/uncertain which potential explanation hypothesis a support instance should be associated with), which results in a combinatorial explosion of possible explanations for a data set. Because of the combinatorics of their answer spaces, interpretation problems require constructive problem solving techniques, and this greatly complicates the complexity of computing the *most probable explanation* (MPE).

4 General Intellectual Issues

While our research is being carried out within the context of distributed sensor interpretation/DSA, many of the issues being addressed are applicable to other important applications. For example, situation assessment is closely related to diagnosis, so we expect that results from our research would apply to distributed diagnosis in complex networks such as LANs and telephone systems. Because our framework is not limited to classification-type problem solving, it should be able to understand and assess more complex situations than the existing systems for these applications. For example, this framework can support the use of active sensors that can be controlled/tuned to best cope with the current situation and provide data that can resolve the most critical uncertainties. There is also increasing interest in the development and use of sophisticated automated/autonomous systems such as robots and autonomous vehicles. Such systems will have to use powerful signal processing and visual systems, and their capabilities should be able to be significantly enhanced if they are provided with the ability to construct a shared view of the situation.

Another example of an increasingly important application where the issues we are exploring are applicable is cooperative information gathering (CIG). Interest in CIG is a result of the recent proliferation of electronically available information (e.g., via the Internet). As a result, information relevant to a user query may be distributed over a large set of sites, and locating the information that provides the best response may require a significant search process. Clearly, distributed search and retrieval has potential advantages in such a situation. However, this process can be very complex since users will have cost and time limits for answering queries, there will be numerous trade-offs involving the cost and speed of accessing particular databases, and there may be differing costs/delays and confidence in the quality of answers depending on agent/search configurations. In addition to the applications discussed above, the research we have performed addresses a range of issues that are of general intellectual interest to the distributed problem-solving community:

- How does the notion of the *bounded rationality* of agents affect coordination strategies, satisficing control strategies, and the quality of solutions;
- How do you make use of sophisticated agents that have a very wide range of possible responses to a situation—e.g., many ways that they could be asked and that they could decide to respond to a request for information;
- What issues arise in reasoning about the termination of problem solving when using a

satisficing approach to control under real-time constraints;

- How does one coordinate very dynamic systems—i.e., where they can be great variance/ uncertainty in the estimates of the quality and duration of potential tasks;
- What implications do the use of complex and dynamically determined information sharing protocols have for the coordination of agents? While the Partial Global Planning mechanism broke new ground in coordinating the local problem solving of agents, the agents it used were very predictable since they had simple and static information sharing protocols. Likewise, the Generalized Partial Global Planning mechanism has assumed fairly static task structures and agents with a limited range of responses.
- What is the appropriate role for meta-level reasoning for coordination in real-time situations;
- Can we now fully realize the potential of the Functionally Accurate/Cooperative (FA/C) distributed problem solving paradigm? The FA/C paradigm involves agents sharing information to make up for incomplete, uncertain, and incorrect data. While previous work has showed that this approach can be used to deal successfully with incomplete data, little was done with data errors and other faults. This work will allow us to explore the types of faults that can be handled and the cost involved (e.g., in terms of redundant processing).
- What are the implications of a distributed system being embeddedwithin another system to which it must provide results? How can the goals of the external system be integrated with the distributed system and how does this affect coordination strategies?
- How do real-time constraints affect cooperation/coordination strategies in distributed problem-solving systems whose problems involve complex, interdependent tasks (i.e., subproblem interactions). This is not just a matter of scheduling activities to meet deadlines, but includes satisficing control in which the goals of the agents are modified.
- Can the agent architecture (laid out in Figure 1) serve as the basis of a generic architecture for distributed situation assessment?

5 Summary of Progress to Date

Our progress to this point can be classified into five main areas, in which we have addressed issues of fundamental importance:

- Significant progress was made in understanding how to handle the transmission and use of incomplete information and information at different levels of abstraction, in DSA systems. This has involved extending the agent belief modeling capabilities of DRESUN agents in order to facilitate the integration of such evidence and the ability to locally exploit it. These advances have resulted in additional flexibility in the local and global problem-solving strategies that can be supported by DRESUN agents.
- Though the generic agent architecture is not complete, we believe that we now have an appropriate model, which simply needs some refinement. This model is based on a layered architecture (see Figure 1). It has a well-defined interface between the component responsible for coordination decisions and that responsible for actual interpretation problem solving. It also has the ability to handle real-time constraints via a scheduling module that is relatively independent, with a well-defined interface to the other modules. This agent model has grown out of work on the Generalized Partial Global Planning mechanism, which uses the TAEMS representation of task structures and a design-to-time real-time scheduler. The design-to-time scheduling approach seems appropriate for

use in DSA since it handles the availability of multiple methods of varying quality to accomplish tasks, and the trade-offs in a satisficing approach to control under real-time constraints.

- We have also expended a significant amount of effort to improve the DRESUN testbed. In part this has involved improvments to the implementation and its interface to facilitate experimentation, and porting the testbed it to modern workstation machines. It has also involved work to complete and/or extend the RESUN/DRESUN model to enhance and explore its suitability as the basis for a generic DSA architecture. Particular areas of development have been:

  - identifying canonical evidence combination methods for the evidence summarization process;
  - representing negative evidence uncertainty in the evidence summaries;
  - representation and summarization of approximate knowledge;
  - additional control plan sequencing constructs;

- We have pursued several lines of research involving the formal characterization and analysis of DSA systems. The need for such tools has become clear as we have addressed problems of increasing complexity. Eventually we expect that these tools will allow us to formally analyze the performance of sophisticated DSA systems in complex environments. Three basic directions are being pursued:

  - Developed analysis tools for understanding coordination issues in DSA systems;
  - Developed an analysis and simulation framework for determining the complexity of interpretation problems;
  - Started to formally analyze the solution quality of distributed interpretation systems.

- We have developed techniques for on-line learning of coordination strategies. This has been a new area of research that was not anticipated when the work was orginally proposed. However, just as the increasing problem and system complexity has motivated use to pursue formal analysis techniques, it has also convinced us that learning has an important role to play in the development and evoluation of coordination strategies, system performance cannot be fully anticipated.

More detailed descriptions of our research progress are contained in the following sections which describe our results in agent modeling in DRESUN; real-time coordination; the formal anlysis of coordination, system solution quality, and problem complexity; and on-line learning of coordination strategies.

6 Agent Modeling in DRESUN

One of the major areas of effort in developing the DRESUN testbed has been to address issues related to modeling the beliefs of other agents. Modeling is important because DSA agents typically must share information in order to satisfy their local goals as well as the overall system goals (since agent subproblems are interdependent). DSA tasks can present several sources of difficulty for information sharing: agents' local evidence may lead to solutions that are globally inconsistent; agent beliefs (interpretations of local data) are uncertain and imprecise; interpretations are complex structures; and beliefs are constantly being revised due to new data and further processing. When an agent shares information about its interpretations with another agent, that information necessarily produces evidence

in a sensor interpretation framework as an integral part of the process of using the information—e.g., checking whether the information is consistent or inconsistent with the local interpretations. Interpretation evidence based on information shared by another agent is referred to as *external evidence.*

Our initial experiments with DRESUN showed that extensions to the model of external evidence were necessary to effectively utilize inter-agent communication of incomplete and conflicting evidence, and evidence at multiple levels of abstraction. The focus of these extensions has been: representing the uncertainties that occur when DRESUN agents exchange such information, providing the ability to reformulate hypotheses to efficiently pursue alternative interpretations, and allowing numeric evaluation of the satisfaction of global termination criteria. The extensions enhance the flexibility of the agents by improving their ability to evaluate the current state of agent beliefs, make better (local) use of incomplete information from another agent, and determine precisely what information is needed to resolve global inconsistencies. This effort turned out to be much more involved than we had orginally forseen.

Among the key modeling issues that we have addressed, we have provided the ability to:

- link multiple views of a hypothesis based on external evidence at different levels of abstraction;
- link multiple hypothesis extensions that are being used in alternative local interpretations (e.g., different portions of a single external vehicle track hypothesis);
- create alternative local versions of external hypotheses based on incomplete information and represent the resulting uncertainties;
- reformulate hypotheses for more efficient exploration and representation of alternatives;
- communicate back results of integrating information that was sent by another agent;
- avoid circular reasoning when exchanging evidence among agents;
- identify when shared information should be updated.

Appendix A contains further elaboration of these ideas.

7 Real-Time Coordination

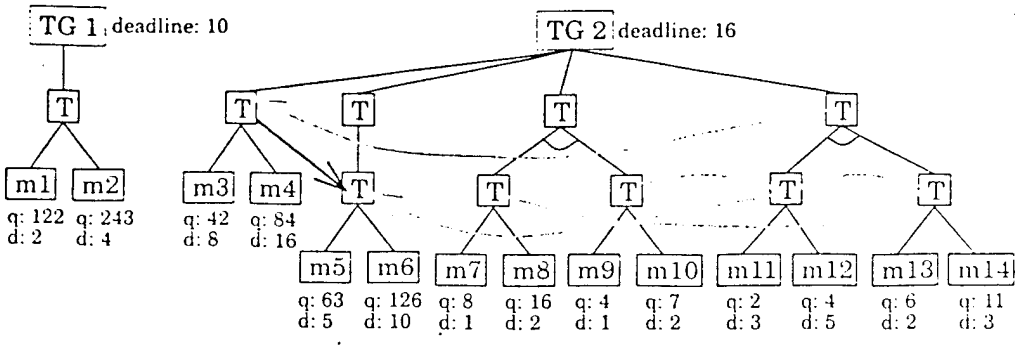


Figure 4: An example task structure with 2 task groups.

The black lines indicate subtask relationships. The thick dark grey line is an enables constraint. The thick light grey lines are facilitates constraints, which represent that there are other tasks whose results will improve the quality of a task's result and or the time to produce the result.

The kinds of complexities that appear in real-time interpretation problems have motivated a new framework for controlling real-time problem solving that we call design-to-time real-time scheduling. In this framework multiple methods are available for many tasks that allow computation time and value (precision, completeness, and certainty) to be traded-off to achieve acceptable results even with tight deadlines for tasks. Design-to-time can be contrasted with the anytime algorithm approach where instead of a set of methods for completing a task there is a single anytime algorithm that achieves increased value as it is given increased computation time. We have found that multiple methods better model the kinds of tasks that are likely to appear in sophisticated interpretation systems. An example of a task structure with multiple methods is given in Figure 4. A schedule for this example problem is given in Figure 5.

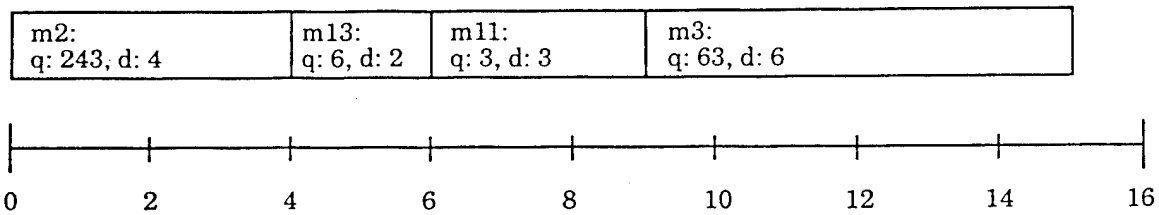


Figure 5: The best schedule for the task structure given in Figure 1.
Note that $m11$ and $m3$ are both facilitated, resulting in higher quality and (in the case of $m3$) lower duration values than given in their initial specification. The total quality of this schedule is 306.

One major focus of the design-to-time work has been on taking interactions among tasks into consideration when scheduling. For example, Task A might facilitate Task B if Task A performs some work that allows Task B to be performed more quickly or with higher value. Because the design-to-time scheduling problem is NP-Hard (in fact, of complexity O(m!) where m is the number of executable methods) most of our work has used heuristic scheduling algorithms.

Recent design-to-time work has focused on negotiation in a multi-agent problem solver with design-to-time scheduling agents. This work has been part of the development of our layered agent architecture (see Figure 1). We have focused on the interface between design-to-time agents and local problem solvers that communicate and coordinate with other problem solvers. One important part of this work is the use of commitments where one agent commits to providing a particular result by a particular time to assist another agent. Schedulers need to balance local problem solving concerns with concerns about providing timely results to other agents. Appendix B contains further elaboration of these ideas.

8 Formal Analysis of Coordination

A key issue in the design of distributed sensor networks is the *organization* of the agents: the extent of each agent's area of responsibility and the amount of *overlap* among these areas. Overlap is an important issue because while it allows for redundancy that can make up for poor quality data and sensor/node failure, processing can become very inefficient if it is not limited. One aspect of our formal work has been to understand the effects of different organizations and coordination approaches.

Our analysis initially focused on exactly what *a priori* knowledge agents had about the distribution of task groups in an episode. First we looked at the distribution of the lowest-

level sensor subtasks of a single task group among multiple agents (deriving the maximum expected number of subtasks), and then we looked at the distribution of task groups themselves. These results were then used to derive the total amount of work, and therefore

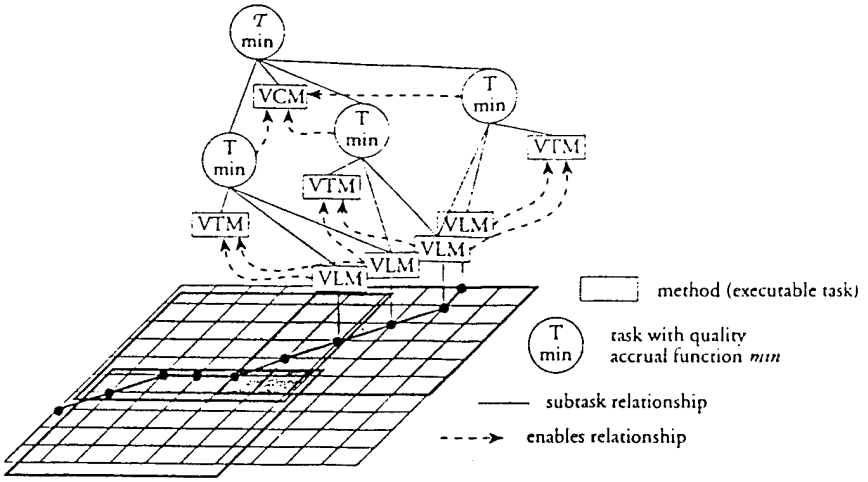


Figure 6: Objective task structure associated with a single vehicle track.

expected termination performance, under various organizational structures and control schemes. The derivation was as follows: starting with the expected number of low-level sensor subtasks at the most heavily-loaded agent we derived the number of task groups at the most heavily-loaded agent given the number of agents that see a single task group and the environmental parameters. We then derived the number of agents that will see a single task group, and the structure of a task group (Figure 6), which allows us to derive the amount of work an agent must do given the number of sensor subtasks it executes, the number of task groups at the agent, and the number of agents involved in a single task group. All of these parts together allow us to derive the expected termination time of a particular system of agents.

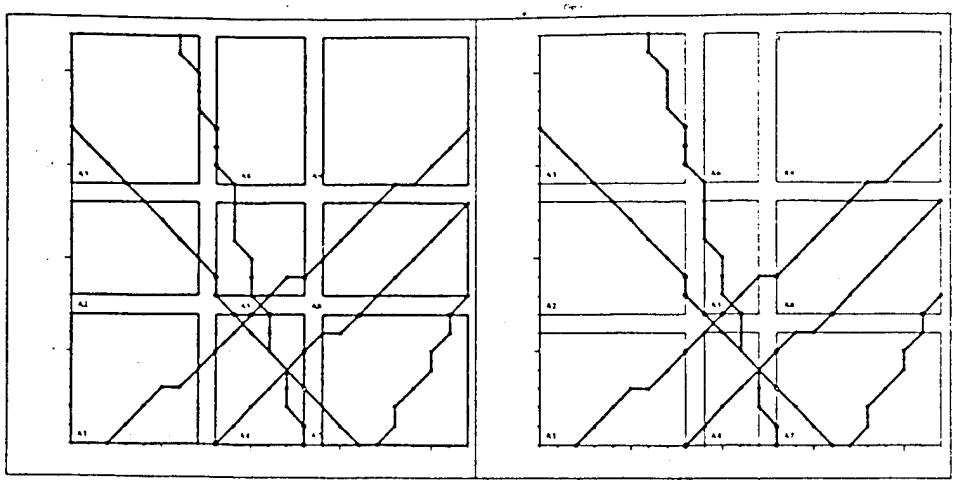


Figure 7: On the left is a 3x3 static organization, on the right is the dynamic reorganization result after agents 3, 4, 5 and 7 attempt to reduce their areas of responsibility by one unit. In this example the corridors running North to South have been moved closer by two units to reduce the load on agents 4, 5, and 6 in the second column.

Our particular implementation of the general idea of dynamically reorganizing the partitions between agents for the DSN simulation keeps each agent's area of responsibility rectangular, and relaxes competing constraints from other agents quickly and associatively (the order of message arrival does not affect the eventual outcome). To do this, the message sent by an agent requests the movement of the four *corridors* surrounding an agent. The northern corridor of Agent 1, for example, is the northern agent organizational responsibility boundary shared by every agent in the same *row* as Agent 1. As can be seen in Figure 7, a 3x3 organization has four corridors (between rows 1 and 2, 2 and 3, and between columns 1 and 2, 2 and 3).

This work has been used to analyze static and dynamic organizational structures for distributed sensor network tasks. We first showed how the performance of any static organization can be statistically described, and then showed under what conditions dynamic organizations do better and worse than static ones. Finally, we showed how the variance in the agents' performance leads to uncertainty about whether a dynamic organization will perform better than a static one given only agent *a priori* expectations. In these cases, we showed when meta-level communication about the actual state of problem solving will be useful to agents in constructing a dynamic organizational structure that outperforms a static one. Viewed in its entirety, our work also presented a methodology for answering questions about the design of distributed problem solving systems by analysis and simulation of the characteristics of a complex environment rather than by relying on single-instance examples. Appendix C further elaborates on these ideas.

9 Formal Analysis of DSA Solution Quality

While several systems that use the FA/C approach have been built, there has never been any formal analysis of the quality of the solutions that are produced by the approach or of the conditions that are necessary for the approach to be successful. We have recently taken a first step toward being able to formally analyzing the quality of solutions that can be produced by an FA/C system. Two theorems have been proven that compare the quality of solutions produced by a distributed system to the solutions that would be produced by an equivalent centralized system. The theorems relate solution quality to agent problem-solving and coordination strategies. They show that there are conditions under which it is possible to guarantee that the distributed system produces a solution that is comparable to the centralized solution, and other conditions under which there is merely some probability of obtaining such a solution. The analysis assumes the capabilities of an abstract model of the DRESUN system for distributed sensor interpretation.

This work is different from most DAI work dealing with global consistency of agent beliefs, which has focused on methods for automatically maintaining (some level of) consistency and has used justification-based representations of belief (e.g., TMSs). Here, we are interested in analyzing the consequences of not maintaining consistency (even within individual agents) in terms of its effect on solution quality. This is an important issue for problems like distributed sensor interpretation where communication and computation considerations often make it impractical to maintain complete consistency (and where consistency is not always necessary for determining an acceptable solution).

We believe that this line of research will help our understanding of the assumptions that underlie the FA/C model of distributed problem solving and that it will assist in producing appropriate propagation and coordination strategies for distributed interpretation systems. It should also make it possible to analyze FA/C frameworks like DRESUN to determine whether the framework is effective in terms of making it possible to use all of the information that the (global) system has available or not (and what the effect there is in not

being able to use all of this information). We are currently working to extend our analyses in several directions: deriving specific probabilities of interest in connection with the theorems as a function of domain models and acceptance thresholds; understanding how to formalize the effects of different evidence propagation and coordination strategies; extending the analyses to deal with processing of only subsets of the available data; and generalizing to CDPS tasks other than sensor interpretation. Appendix D contains further elaboration of these ideas.

10 Analysis of the Complexity of Interpretation Problems

The *IDP/UPC* formalism was developed for analyzing the relationship between the performance of search-based interpretation problem solving systems and the inherent properties, or *structure*, of problem domains in which they are applied. The formalism models interpretation *domain theories* (i.e., the computational theory that is the basis for a problem solver's functionality) in terms of four feature structures: *component* (or *syntax*), *utility* (or *credibility*), *probability* (or *distribution*), and *cost*. The different feature structures are represented in terms of a *domain grammar* and functions associated with production rules of the grammar. Search paths are represented graphically as *interpretation trees* of the grammar. By analyzing the statistical properties of the interpretation trees of a domain grammar, it is possible to determine characteristics of problem solving such as the expected cost for a random problem instance and the expected credibility of a solution.

Figure 8 shows an example of a domain grammar used for analyzing sensor interpretation tasks in a vehicle tracking domain. Figure 9 presents an example problem instance generated with the domain grammar.

The most important aspect of the IDP/UPC formalism is that it supports a unified representation of both *meta-operators* and domain processing. This unified representation is based on viewing sophisticated control architectures as mechanisms that evaluate problem solving actions by selectively applying a process that examines a search path's relationship with other, possibly interacting, search paths. (Relationships are based on the distribution of domain events and are statistical in nature. For example, a relationship might indicate that two search paths lead to the same final result 50% of the time.) Furthermore, the process of examining a search path's relationships are viewed as a *distinct search operation* and not as part of the control architecture. Interrelationships between search paths are represented as abstract or approximate states that are explicitly created by search operators and not by a monolithic control process.

The IDP/UPC formalism was developed to analyze the assumptions (implicit and explicit) that meta-operators make about a domain. This work represents an initial step needed to formalize complex distributed search processes that occur in DSA systems. Appendix E further elaborates on these ideas.

11 On-Line Learning of Coordination Strategies

Although coordination is an essential technique for the design of DSA systems, sophisticated coordination strategies are not always cost-effective in all problem-solving situations, nor is it always possible to understand even after the DSA system is operational all the task and environmental variables that come into play in deciding what form of coordination at any specific instance is most appropriate. Thus, we have been investigating an on-line learning method to acquire coordination plans for specific problem-solving situations so that the appropriate type of coordination strategy is used. This effort was not anticipated in the original proposed scope of work but as we have come to understand the

Sample Grammar for Complex Vehicle Tracking Domain

1. $*S_{[f]} \rightarrow Noise_{[f]}\ Tracks_{[f]}$
2. $Noise_{[f,t]} \rightarrow Noise_{[f,t+1]}\ N_{[f,t]}$

3. $N_{[f,t]} \rightarrow N_{[f,t]}\ n_{[f,t]}$ p = .10
   $\rightarrow n_{[f,t]}$ p = .25
   $\rightarrow \lambda$ p = .65

4. $Tracks_{[f]} \rightarrow Tracks_{[f]}\ Track_{[f]}$ p = .10
   $\rightarrow Track_{[f]}$ p = .90

5. $Track_{[f,t,x,y]} \rightarrow I\text{–}Track_{[f,t,x,y]}$ p = .50
   $\rightarrow P\text{-}Track_{[f,t+1,x+vel+acc,y+vel+acc]}\ P\text{-}T_{[f,t,x,y]}$ p = .25
   $\rightarrow G\text{-}Track_{[f,t+1,x+vel+acc,y+vel+acc]}\ G\text{-}T_{[f,t,x,y]}$ p = .25

6. $I\text{-}Track_{[f,t,x,y]} \rightarrow I\text{–}Track_{[f,t+1,x+vel+acc,y+vel+acc]}\ T_{[f,t,x,y]}$

7. $P\text{-}Track \rightarrow P\text{-}Track_{[f,t+1,x+vel+acc,y+vel+acc]}\ P\text{-}T_{[f,t,x,y]}$

8. $G\text{-}Track \rightarrow G\text{-}Track_{[f,t+1,x+vel+acc,y+vel+acc]}\ G\text{-}T_{[f,t,x,y]}$

9. $P\text{-}T_{[f,t,x,y]} \rightarrow T_{[f,t,x+offset,y+offset]}\ T_{[f,t,x,y]}$

10. $G\text{-}T_{[f,t,x,y]} \rightarrow G_{[f,t,x+offset,y+offset]}\ T_{[f,t,x,y]}$

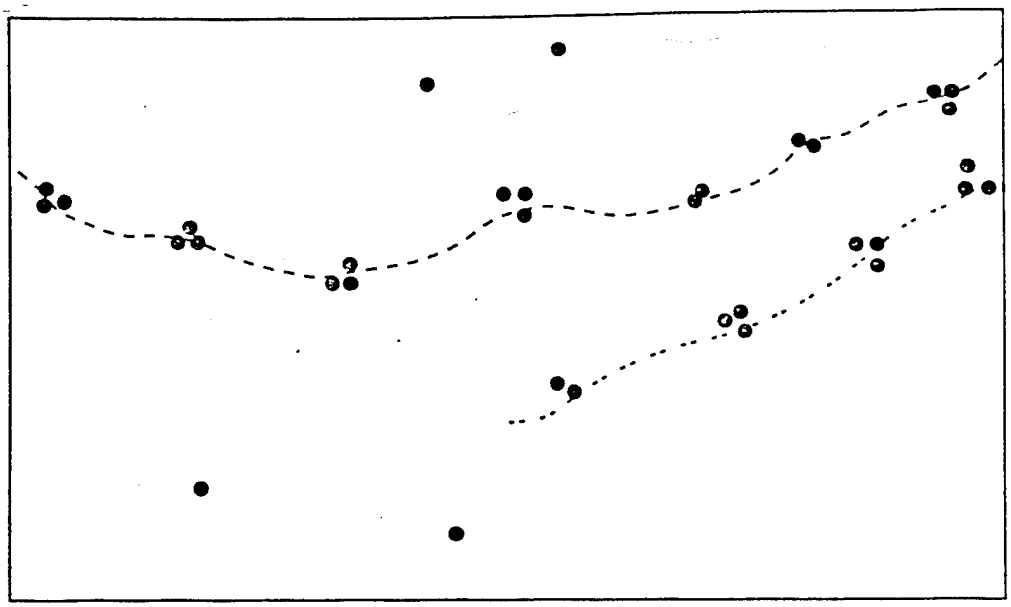
11. $T_{[f,t,x,y]} \rightarrow V_{[f,t,x,y]}\ Correlated\text{-}Noise(?)_{[f,t,x,y]}$

12. $V_{[f,t,x,y]} \rightarrow GC1_{[f,t,x,y]}\ GC3_{[f,t,x,y]}\ GC7_{[f,t,x,y]}$ p = .40
    $\rightarrow GC1_{[f,t,x,y]}\ GC3_{[f,t,x,y]}$ p = .30
    $\rightarrow GC1_{[f,t,x,y]}\ GC7_{[f,t,x,y]}$ p = .25
    $\rightarrow \lambda$ p = .05

13. $G_{[f,t,x,y]} \rightarrow G\text{-}GC1_{[f,t,x,y]}\ G\text{-}GC3_{[f,t,x,y]}\ G\text{-}GC7_{[f,t,x,y]}$ p = .70
    $\rightarrow G\text{-}GC1_{[f,t,x,y]}\ G\text{-}GC3_{[f,t,x,y]}$ p = .20
    $\rightarrow G\text{-}GC1_{[f,t,x,y]}\ G\text{-}GC7_{[f,t,x,y]}$ p = .05
    $\rightarrow \lambda$ p = .05

* The subscript notation in brackets indicates *feature list* information where the symbol f indicates all feature list list information not represented explicitly.

Figure 8: An Example Grammar for a Vehicle Tracking/Interpretation Domain



● : signal data from sensor. Each point has a time, location, energy level, and frequency.

: the actual path of the vehicle that generated the data.

: a ghost image of the actual path.

Figure 9: Problem Instance Generated with Vehicle Tracking Grammar

design requirements of a DSA system we realized the key role such technology could play. This learning is accomplished by recording and analyzing traces of inferences after problem solving. The analysis identifies situations where inappropriate coordination strategies have resulted in redundant activities or the lack of timely execution of important activities, thus degrading system performance. The analysis is also used to create situation-specific coordination plans that use additional non-local information about activities in the networks, which are added to the system to remedy the problem. These situation-specific plans can decide (1) priorities of local and non-local activities, (2) priorities of messages, and (3) what non-local information is needed for coordination activities. These plans also have the effect of introducing different levels of coordination into this system, that is, the system can decide when tight coordination is required and when autonomous processing is more appropriate. This project investigates this approach using a distributed computer-network monitoring and diagnosis system. Appendix F contains further elaboration of these ideas.

In summary, we feel that results of the research supported under this contract have contributed significantly to the eventual construction of practical Distributed Situation Assessment systems.

12 PUBLICATIONS, REPORTS AND ARTICLES

12.1 Refereed Papers Accepted but not yet Published

NagendraPrasad, M.V., Lesser, V. and Lander, S. "Reasoning and Retrieval in Distributed Case Bases," to appear in *Journal of Visual Communication and Image Representation, Special Issue on Digital Libraries.*

12.2 Refereed Papers Published

Bhandaru, M. and Lesser, V. "Discrepancy Directed Model Acquisition for Adaptive Perceptual Systems." *IJCAI-95 Workshop on Computational Auditory Scene Analysis*, Montreal, Canada, 1995.

Bhandaru, M., Draper, B. A., and Lesser, V., "Learning Image to Symbol Conversion," AAAI 1993 Fall Symposium Series: Machine Learning in Computer Vision: What, Why and How? Raleigh, North Carolina, October 1993.

Carver, N. and Lesser, V. "The DRESUN Testbed for Research in FA/C Distributed Situation Assessment: Extensions to the Model of External Evidence," *Proceedings of the First International Conference on Multi-Agent Systems*, June 1995.

Carver, N. and Lesser, V., "The Evolution of Blackboard Control Architectures in *Expert Systems with Applications—Special Issue on the Blackboard Paradigm and Its Applications.* Vol. 7, pp. 1–30, 1994.

Carver, N. and Lesser, V. "Resolving Global Inconsistency in Distributed Sensor Interpretation: Modeling Agent Interpretations in DRESUN," *Proceedings of Twelfth International Workshop on Distributed Artificial Intelligence*, 1993.

Decker, K. and Lesser, V. "Coordination Assistance for Mixed Human and Computational Agents." In *Proceedings of the Second International Conference on Concurrent Engineering Research and Applications*, McLean, Virginia, August 1995.

Decker, K. and Lesser, V. "Designing a Family of Coordination Algorithms," *Proceedings of the First International Conference on Multiagent Systems*, San Francisco, June 1995. AAAI Press.

Decker, K.S. and Lesser, V.R. "Designing a family of coordination algorithms" in the *Proceedings of 13th International Distributed Artificial Intelligence Workshop*, July 1994.

Decker, K.S. and Lesser, V.R. "Communication in the Service of Coordination," in the *Proceedings of the Workshop on Planning for Interagent Communication*, AAAI, Seattle, WA, July 1994.

Decker, K. and Lesser, V. "Task Environment Centered Design of Organizations" in *Computational Organization Design, Working Notes of the AAAI Spring Symposium*, Ingemar Hulthage (ed.), 1994.

Decker, K. and Lesser, V., "Examples of Quantitative Modeling of Complex Computational Task Environments," Workshop on AI and Theories of Groups & Organizations: Conceptual and Empirical Research, AAAI-93, Washington, DC, 1993.

Decker, K. and Lesser, V.R. "Quantitative Modeling of Complex Environments," *International Journal of Intelligent Systems in Accounting, Finance and Management, special issue on Mathematical and Computational Models of Organizations: Models and Characteristics of Agent Behavior.* Vol. 2: 215-234, 1993.

Decker, K. and Lesser, V., "Analyzing a Quantitative Coordination Relationship," *Group Decision and Negotiation*, 2:195–217, 1993.

Decker, K. and Lesser, V. "An Approach to Analyzing the Need for Meta-Level Communication," *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, 1993.

Decker, K. and Lesser, V. "A One-Shot Dynamic Coordination Algorithm for Distributed Sensor Networks," *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pp. 210–216, 1993.

Decker, K. and Lesser, V. "Quantitative Modeling of Complex Computational Task Environments," *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pp. 217–224, 1993.

Decker, K. and Lesser, V., "Quantitative Modeling of Complex Computational Task Environments," *Twelfth International Workshop on Distributed Artificial Intelligence*, 1993.

Decker, K. and Lesser, V., "Analyzing the Need for Meta-Level Communication," *Twelfth International Workshop on Distributed Artificial Intelligence*, 1993.

Decker, K. S., Garvey, A.J., Lesser, V.R., and Humphrey, M.A., "An Approach to Modeling Environment and Task Characteristics for Coordination," *AAAI Workshop on Enterprise Integration*, San Jose, July 1992.

Decker, K. and Lesser, V., "Generalizing The Partial Global Planning Algorithm," *International Journal on Intelligent Cooperative Information Systems,* 1(2):319-346, 1992.

Garvey, A. and Lesser, V. "Design-to-time Scheduling and Anytime Algorithms." *Proceedings of the Workshop on Anytime Algorithms and Deliberation Scheduling*, IJCAI-95, Montreal, Canada.

Garvey, A. and Lesser, V., "A Survey of Research in Deliberative Real-Time Artificial Intelligence," *The Journal of Real-Time Systems*, 6(3):317-347, May 1994.

Garvey, A. and Lesser, V., "Research Summary of Investigations Into Optimal Design-to-time Scheduling," in *Proceedings of AAAI Workshop on Experimental Evaluation of Reasoning and Search Methods,* Seattle, WA, July 1994.

Garvey, A., Decker, K. and Lesser, V. "A Negotiation-based Interface Between a Real-time Scheduler and a Decision-Maker," in *Proceedings of Workshop on Models of Conflict Management in Cooperative Problem Solving*, AAAI, Seattle, WA, July 1994.

Garvey, A., Humphrey, M., and Lesser, V. "Task Interdependencies in Design-to-time Real-time Scheduling," *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pp. 580–585, 1993.

Garvey, A. and Lesser, V. "Scheduling Satisficing Tasks with a Focus on Design-to-time Scheduling," *Proceedings of IEEE Workshop on Imprecise and Approximate Computation*, Phoenix, AZ, pp. 25-29, December 1992.

Klassner, F., Lesser, V. and Nawab, S.H. "The IPUS Blackboard Architecture as a Framework for Computational Auditory Scene Analysis," 1995 IJCAI Workshop on Computational Auditory Scene Analysis, Montreal, Canada, August 19-20, 1995.

Lesser, V., Nawab, S.H., and Klassner, F. "IPUS: an architecture for the integrated processing and understanding of signals," in *Artificial Intelligence*, 77 (1995) 129–171.

Oates, T., Nagendra Prasad, M.V., Lesser, V. and Decker, K. "A Distributed Problem Solving Approach to Cooperative Information Gathering" in *AAAI-95 Spring Symposium on Information Gathering from Heterogeneous, Distributed Environments*, Stanford University, Stanford, CA, March, 1995.

Oates, T., Nagendra Prasad, M.V. and Lesser, V. "Networked Information Retrieval as Distributed Problem Solving," *Proceedings of the CIKM Workshop on Intelligent Information Agents*, NIST, Gaithersburg, Maryland, December 1994.

Sugawara, T. and Lesser, V. "Learning Coordination Plans in Distributed Problem-Solving Environments." Abstract in *Proceedings of the First International Conference on Multi-Agent Systems*, San Francisco, June 1995. AAAI Press.

Sugawara, T. and Lesser, V., "On-Line Learning of Coordination Plans," *Twelfth International Workshop on Distributed Artificial Intelligence*, 1993.

12.3 Unrefereed Reports and Articles:

Carver, N. and Lesser, V., "The Evolution of Blackboard Control Architectures," Computer Science Technical Report 92–71, University of Massachusetts, 1992. (This is a revised and extended version of paper with same title to appear in *Expert Systems with Applications—Special Issue on the Blackboard Paradigm and Its Applications.*)

Carver, N. and Lesser, V. "Distributed Sensor Interpretation: Modeling Agent Interpretations in DRESUN," Computer Science Technical Report 93–75, University of Massachusetts, Amherst, 1993. (This is a revised and extended version of that which appeared in *Twelfth International Workshop on Distributed Artificial Intelligence*, 1993.)

Decker, K. and Lesser, V. "Coordination Assistance for Mixed Human and Computational Agents." Computer Science Technical Report 95–31, University of Massachusetts, 1995.

Decker, K. and Lesser, V.R. "Designing a Family of Coordination Algorithms," Computer Science Technical Report 94–14, University of Massachusetts at Amherst, January 1994.

Decker, K. and Lesser, V. "Quantitative Modeling of Complex Computational Task Environments," Computer Science Technical Report 93–21, University of Massachusetts, 1993.

Decker, K. and Lesser, V., "Analyzing the Need for Meta-Level Communication," Computer Science Technical Report 93–22, University of Massachusetts, 1993.

Decker, K. and Lesser, V., "Examples of Quantitative Modeling of Complex Computational Task Environments," Workshop on AI and Theories of Groups & Organizations: Conceptual and Empirical Research, AAAI-93, Washington, DC, 1993.

Garvey, A. and Lesser, V. "Design-to-time Scheduling With Uncertainty," University of Massachusetts Computer Science Department Technical Report 95–03, 1995.

Garvey, A., Decker, K. and Lesser, V. "A Negotiation-based Interface Between a Real-time Scheduler and a Decision-Maker," Computer Science Technical Report 94–08, University of Massachusetts at Amherst, January 1994.

Garvey, A. and Lesser, V., "A Survey of Research in Deliberative Real-Time Artificial Intelligence," Computer Science Technical Report 93–84, University of Massachusetts, November 1993.

Garvey, A. and Lesser, V., "Scheduling Satisficing Tasks with a Focus on Design-to-time Scheduling," Proceedings of IEEE Workshop on Imprecise and Approximate Computation, Phoenix, AZ, pp. 25-29, December 1992.

Klassner, F. and Mani, R., "The IPUS C.2 Sound Understanding Testbed Acoustic Modeling Framework and Sound Library," University of Massachusetts Computer Science Department Technical Report 95-65.

Nagendra Prasad, M., Lesser, V., and Lander, S. "Retrieval and Reasoning in Distributed Case Bases." Computer Science Technical Report 95–27, University of Massachsuetts at Amherst, March 1995.

Oates, T., Nagendraprasad, M. and Lesser, V. "Cooperative Information Gathering: A Distributed Problem-Solving Approach," Computer Science Technical Report 94-66, University of Massachusetts, August 1994.

Sugawara, T. and Lesser, V., "On-Line Learning of Coordination Plans," Computer Science Technical Report 93–27, University of Massachusetts, 1993.

Whitehair, R. C. and Lesser, V.R. "A Framework for the Analysis of Sophisticated Control in Interpretation Systems," Computer Science Technical Report 93–53, University of Massachusetts, 1993.

Whitehair, R. C. and Lesser, V.R., "A Methodology for Modeling Sophisticated Problem Solvers," UMASS Technical Report 95-39, 1995.

Whitehair, R. C. and Lesser, V.R., "IDP: A Formalism for Modeling and Analyzing Complex Interpretation Problem Domains," UMASS Technical Report 95-37, 1995.

12.4 Books or Parts Thereof Published

Decker, K. S., Garvey, A.J., Lesser, V.R., and Humphrey, M.A., "An Approach to Modeling Environment and Task Characteristics for Coordination," in *Enterprise Integration Modeling*: *Proceedings of the First International Conference,* Charles J. Petrie, Jr. (ed.). Cambridge: MIT Press, 1992, pp. 379–388.

Garvey, A. and Lesser, V., "Representing and Scheduling Satisficing Tasks," in *Imprecise and Approximate Computation,* S. Natarajan (ed.). Norwell, MA: Kluwer Academic Publishers, pp. 23-34, 1995.

12.5 Books or Parts Thereof Accepted but not yet Published

Decker, K.S., "Distributed Artificial Intelligence Testbeds," to appear in *Foundations of Distributed Artificial Intelligence*, Wiley Inter- Science, Chapter 5, G. O'Hare and N. Jennings (eds.).

Decker, K.S., "TAEMS: A framework for analysis and design of coordination mechanisms," to appear in *Foundations of Distributed Artificial Intelligence*, Wiley Inter-Science, Chapter 17, G. O'Hare and N. Jennings (eds.).

12.6 Ph.D. Dissertations

Decker, Keith S. "Environment Centered Analysis and Design of Coordination Mechanisms," Ph.D. Dissertation and Computer Science Technical Report 95–69, University of Massachusetts at Amherst, May 1995.

13 TECHNOLOGY TRANSFER

A distributed local-area network diagnosis system has been developed by NTT that incorporates the techniques for on-line learning of coordination strategies developed under this contract.

The RESUN architecture has been reimplemented in C++ by Professor Hamid Nawab of Boston University to be used as part of a radar signal processing testbed.

The RESUN interpretation system is being used by Professor Paul Cohen of University of Massachusetts in his development of an intelligent assistant for data analysis.

APPENDIX A

ICMAS-95

The DRESUN Testbed for Research in FA/C Distributed Situation Assessment: Extensions to the Model of External Evidence*

Norman Carver
Department of Computer Science
Southern Illinois University at Carbondale
Carbondale, IL 62901
(carver@cs.siu.edu)

Victor Lesser
Department of Computer Science
University of Massachusetts
Amherst, Massachusetts 01003
(lesser@cs.umass.edu)

Abstract

This paper reports on extensions that have been made to the DRESUN testbed for research on distributed situation assessment (DSA). These extensions involve issues that have arisen in modeling the beliefs of other agents when dealing with inter-agent communication of incomplete and conflicting evidence, and evidence at multiple levels of abstraction. The extensions support highly directed exchanges of evidence among agents because they better represent the uncertainties that occur when DRESUN agents exchange incomplete and conflicting information. This is important in FA/C systems because agents must share results in order to satisfy their local goals as well as the overall system goals. Thus, sharing must be done efficiently for an FA/C approach to be effective. These issues will arise in any distributed problem solving application involving interacting subproblems, when agents must function without complete and up-to-date information.

Introduction

The *functionally accurate, cooperative* (FA/C) paradigm for distributed problem solving (Lesser & Corkill 1981; Lesser 1991) was proposed for applications in which tasks are naturally distributed but in which the distributed subproblems are not independently solvable. In the FA/C approach, agents produce tentative, partial results based on local information and then exchange these results with the other agents. The constraints that exist among the agents' subproblems are then exploited to resolve the local uncertainties and global inconsistencies that occur due to the lack of accurate, complete, and up-to-date local information.

In (Carver & Lesser 1991b) we described the capabilities of the initial implementation of DRESUN, a testbed for research on distributed situation assessment (DSA)[1] using an FA/C approach. DRESUN was developed to explore the implications of having agents with more sophisticated evidential representations and control capabilities than the agents that were used in earlier research with the Distributed Vehicle Monitoring Testbed (DVMT) (Lesser & Corkill 1983; Durfee & Lesser 1991). Because of agent limitations, that research did not adequately address several important issues that arise when sharing incomplete and inconsistent information among DSA agents. Furthermore, overall agent activities were not driven by an explicit need to produce local solutions that were globally consistent (let alone globally optimal).

This paper reports on extensions to the initial DRESUN testbed, related to modeling the beliefs/evidence of other agents. The basic DRESUN architecture provides a good basis for an FA/C approach because inter-agent subproblem interactions are explicitly represented and used to drive problem solving. However, our experiments showed that extensions to the model of *external evidence*[2] were necessary to make the most effective use of inter-agent communication of incomplete and conflicting evidence, and evidence at multiple levels of abstraction. The focus of these extensions has been on representing the uncertainties that occur when DRESUN agents exchange such information, determining precisely what information is needed to resolve global inconsistencies, and providing the ability to reformulate hypotheses to efficiently pursue alternative interpretations.

These issues are very important in FA/C systems because agents must share results in order to satisfy their local goals as well as the overall system goals.

*This work was supported by the Department of the Navy, Office of the Chief of Naval Research, under contract N00014-92-J-1450. The content of the information does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred.

[1] Situation assessment involves the fusion of sensor data, intelligence information, and so forth, and the interpretation of this information to produce a high-level description of the situation in the environment.

[2] In a DSA framework, results from another agent necessarily produces evidence as an integral part of the process of using the information—e.g., checking whether the information is consistent or inconsistent with the local interpretations. Evidence based on information shared by another agent is referred to as external evidence.

To resolve *data uncertainties*, an agent must be able to evaluate whether other agents' results are consistent or inconsistent with its own results, and integrate these other agents' results to revise and extend its local partial results (Lesser & Corkill 1981). A key assumption of the FA/C approach is that agents can do this without "excessive" communication among the agents.[3] However, DSA tasks can present several sources of difficulty for efficient results sharing: agents' local data may lead to solutions that are globally inconsistent; agent beliefs/results are uncertain and imprecise; results (e.g., interpretations) are complex structures; and beliefs are constantly being revised due to new data and further processing.

The scenario in Figure 1 is an example of a distributed vehicle monitoring situation in which local solutions are inconsistent and extended agent interactions are necessary to resolve the inconsistency. We will use this example to introduce some key issues in results sharing, and we will return to it in more detail in the Resolving Global Inconsistency Section, to explore the representation of external evidence in DRESUN.

In the example, processing of only their own local data would cause agent A and agent B to form the track hypotheses T_a and T_b, respectively. Because the tracks extend through an area of overlapping interest, the agents can recognize that they must communicate to verify the global consistency of their local interpretations. These tracks are inconsistent since they imply that either a single vehicle is in different places at the same time or else two vehicles are in the same place at the same time. One important thing to note here is that while exchanging abstract results (the track hypotheses without their supporting evidential structures) allows the inconsistency to be detected, this level of information is not sufficient to allow the inconsistency to be resolved (i.e., there remains uncertainty about the correct global interpretation). This is because neither T_a nor T_b is significantly more likely than the other since each includes some good quality data and some poor quality data. (Even when partial results are "consistent," uncertainties may result when only abstractions are exchanged—see the Representing External Evidence Section.)

Resolving the inconsistency in favor of the most likely global interpretation requires an understanding of the level of belief provided by data supporting the different portions of each track. In other words, more detailed information about the interpretation hypotheses is required. One way to insure that agents have all the necessary information would be to always communicate the complete evidential structure associated with the hypotheses. However, because this structure can be very complex, communication and processing limitations will typically make it impractical to fully communicate this information.[4] Furthermore, complete communication is often not necessary. For instance, we will see that in this example each agent does not have to have complete, detailed information about each other's raw data in order to resolve the inconsistency.

Thus, what is needed is a system with the flexibility to request or respond with information at different levels of detail—based on the dynamic problem-solving requirements—as part of an incremental process of resolving inconsistency. This requires the ability to integrate incomplete results information, represent the resulting uncertainty, and use this representation to drive further actions. The DRESUN architecture provides these capabilities.

Because of its representation of inconsistency as a source of uncertainty and its emphasis on directed interactions to resolve inconsistency, DRESUN differs from most DAI work dealing with the global consistency of local agent beliefs. This work (e.g., (Courand 1990; Huhns & Bridgeland 1991)) has largely focused on methods for automatically maintaining (some particular level of) consistency and has used justification-based representations of belief (e.g., TMSs). DRESUN does not automatically enforce complete consistency because this can be very expensive both in terms of communication and computation, and it is usually not necessary. DRESUN uses an evidential (partial beliefs) representation rather than a justification-based representation of its beliefs because virtually all evidence and hypotheses in a DSA system are uncertain.

The next section reviews the DRESUN architecture, in which agent interactions are driven by the need to resolve uncertainty about the global consistency of local solutions. This is followed by a section that examines some of the issues that have arisen in representating external evidence in DRESUN. The example introduced in this section is then explored in more detail, and the paper concludes with a brief summary and a discussion

[3] Another current research direction of ours is formal analysis of the FA/C model in terms of the domain characteristics necessary for the approach to be effective and the quality of solutions that can be produced—see (Carver & Lesser 1994; Carver 1995). For instance, effective FA/C problem solving requires one or more of the following: 1.) only a subset of each agent's subproblems interact with those of other agents; 2.) it can be determined what local data/abstractions are relevant to which other agents; 3.) data abstractions (i.e., the tentative, partial results) can substitute for the raw data in determining the global consistency and likelihood of local solutions. If these conditions are not satisfied then a centralized approach may have better performance (though a distributed approach may still be preferred due to factors like tighter coupling of the processor and sensor, or increased reliability and graceful performance degradation).

[4] The example shown here is simplified to allow us to focus on our main points. It shows only a small fraction of the data that would be faced by most real-world DSA systems, and it does not show the numerous alternative interpretation hypotheses that would be interrelated with any solution hypotheses.

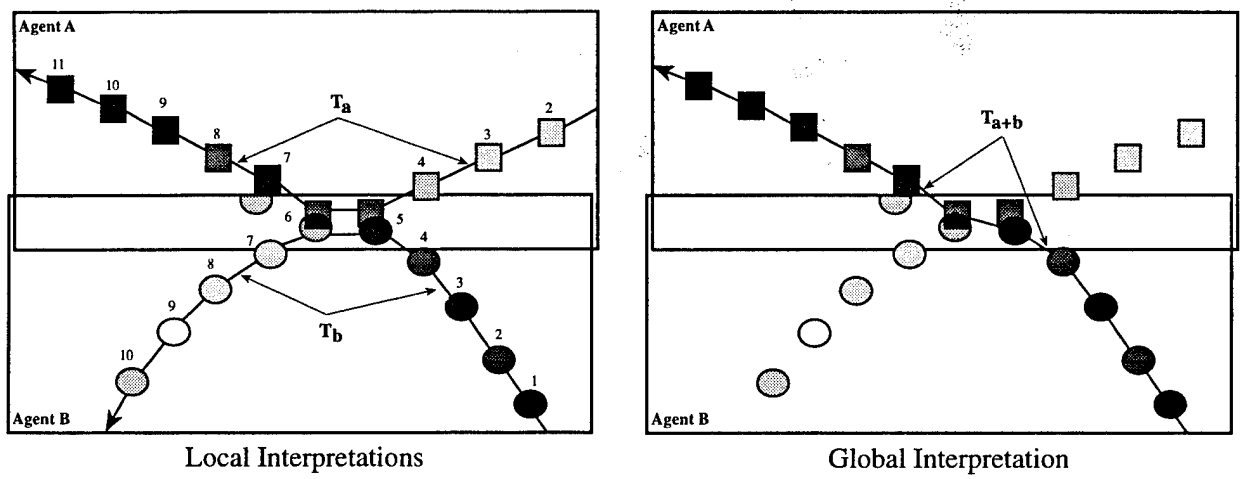


Figure 1: An example of inconsistent local interpretations.
The application is vehicle monitoring. Agent A and Agent B receive data only from their own individual sensors, whose coverage regions overlap. Agent A's data is represented by squares and agent B's by circles, with positions as indicated at the times denoted by the associated numbers. The grey density of the data points corresponds to the relative "quality" of the data—i.e., the a priori likelihood that the data would have resulted from a real vehicle. and the spectral content of the acoustic signals. "Empty" points denote data whose existence has been assumed by the agents. Based on its own data, each agent would form the local interpretations shown: agent A would hypothesize vehicle track T_a and agent would hypothesize vehicle track T_b. T_a covers agent A's data from times 2 through 11, and T_b covers agent B's data from times 1 through 10. The preferred global interpretation—given a complete view of the data from both agent A and agent B—is T_{a+b} because it covers more high quality data than either of the local tracks (the remaining uninterpreted data is due to ghosting phenomena and may or may not be explicitly interpreted depending on the termination criteria of the system). T_{a+b} covers agent B's data from times 1 through 6 and agent A's data from times 5 through 11 (it covers both agents' consistent data at times 5 and 6).

of current research issues.

DRESUN

DRESUN agents are RESUN interpretation systems (Carver & Lesser 1991a; Carver & Lesser 1993). One of the key ideas in RESUN is the use of symbolic *source of uncertainty* statements (SOUs) in the evidence for the interpretation hypotheses. The SOUs allow agents to understand the reasons why their hypotheses are uncertain and why their termination criteria remain unmet. RESUN also supports the use of satisficing control and heuristically controlled, approximate evaluation of hypothesis belief to deal with the computational complexity of DSA problems (Carver & Lesser 1994).

In an FA/C system, there must be some mechanism to drive the exchange of results among the agents so that incorrect and inconsistent local solutions can be detected and dealt with. Ideally, this would be accomplished with a mechanism that allows agents to understand where there are constraints among their subproblems, so that information interchange could be highly directed. DRESUN provides just this capability for DSA applications. DRESUN agents create *global consistency* SOUs whenever it is determined that a local hypothesis can obtain evidence from another agent—i.e., whenever a subproblem interaction (constraint) is detected. These SOUs are viewed as sources of uncertainty about the correctness of an agent's local solution because they represent unresolved questions about the global consistency of the solution. Examples of situations involving each of the global consistency SOUs are shown in Figure 2.

DRESUN's global consistency SOUs make explicit the possible interrelationships between agents' local subproblems, and provide an integrated view (in conjunction with the standard RESUN SOUs) of both the local and global problem-solving goals, which drive agent control decisions. Thus in DRESUN, agents exchange results based on the goal of insuring the global consistency of their local solutions. So far, though, we have not discussed what "resolving" a global SOU means. Resolution of a global SOU involves exchanging information among the associated agents so as to effectively propagate evidence between their hypothesis-evidence networks.[5] An example of the resolution of a global SOU is shown in Figure 3. Resolution of global SOUs is analogous to (intra-agent) evidence propagation, and as with evidence propagation there are a range of strategies that may be used to determine which global SOUs to pursue and how

[5]While there are significant differences between these networks and Bayesian or belief nets, for our purpose here the reader can consider them to be similar.

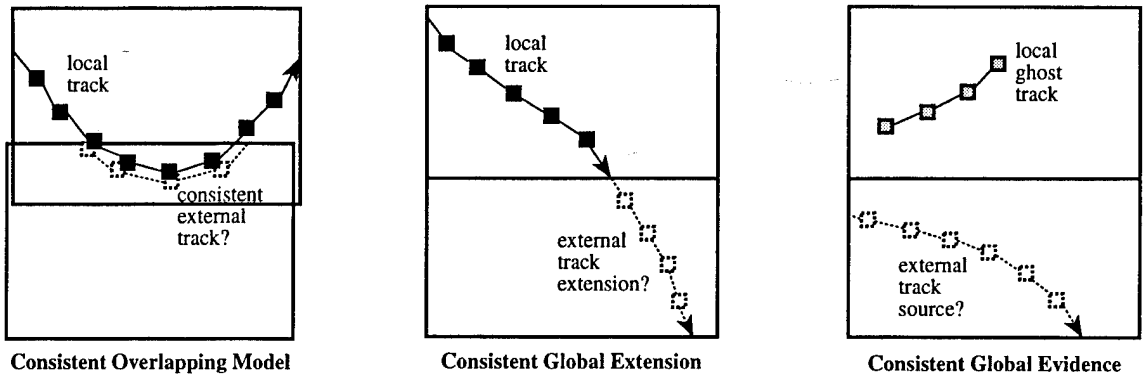


Figure 2: Examples of the global consistency SOUs for vehicle monitoring.
There are three types of global consistency interactions in sensor interpretation problems: interpretations in regions of overlapping interest among agents must be consistent, "continuous" hypotheses (e.g., vehicle tracks) that would extend into other agents' areas must have consistent external extensions, and hypotheses that require evidence that could be in another agent's area (e.g., the source for a ghost track hypothesis) must have appropriate external evidence. Instances of these situations can be detected given the domain model and knowledge of the organization of agent interest areas. DRESUN uses three global consistency SOUs to denote instances of these global interactions: *consistent-overlapping-model*, *consistent-global-extension*, and *consistent-global-evidence*.

completely to propagate their effects. We will not explore the issues involved in these choices here, but see (Carver & Lesser 1994).

Representing External Evidence

The DRESUN architecture provides the basis for effective FA/C problem solving because it explicitly represents the inter-agent subproblems interactions and uses this information to drive the exchange of partial results among the agents. However, our initial experimentation with DRESUN found that there were some restrictions on the coordination strategies that could be supported because of an inability to represent the uncertainties that arise when using incomplete results information and results information at multiple levels of abstraction. This section will try to make clear the differences between representing local evidence and external evidence, by focusing on the problems that arise when evaluating the effect of (both consistent and inconsistent) incomplete external evidence.

First, we must make the concept of global consistency/inconsistency more precise. In interpretation problems, data and hypotheses[6] are consistent if they can be *merged* into a single valid interpretation hypothesis or if they relate only to completely independent top-level hypotheses. For example, two vehicle track hypotheses that overlap in time are consistent if their (imprecise) vehicle type parameters are consistent, if their (imprecise) positions intersect at the overlapping times, and if their positions for the non-overlapping times are consistent with vehicle movement constraints. Consistency checking is straightforward in DRESUN.[7]

When consistent local hypotheses are merged, agents have all the evidence necessary to construct a *complete* new hypothesis. For example, in the consistent local evidence example in Figure 4, the supporting data of T_1 and T_2 can be used to create a new hypothesis T_3. Now, consider the case in which T_2 is an external hypothesis, and the local agent does not have (immediate access to) any of T_2's supporting evidence. In this case, the local agent can also create a new hypothesis T_3, which has the same attributes (e.g., positions and vehicle ID) as the T_3 created in the local evidence case. However, without access to the evidence for the external T_2, the belief in this T_3 cannot be properly evaluated. Evaluating the belief in T_3 requires knowledge of the quality of the data for each of supporting vehicle positions, but all the local agent has access to is the overall belief in T_2—which depends on the quality of the data from the overlapping positions as well as the positions that extended T_1. While the belief in T_3 might be estimated from this evidence (assuming, for instance, that T_2's overlap data is of about the same quality as T_1's), the resulting belief rating will be uncertain.

One of the characteristics that makes sensor interpretation difficult is that inconsistency (i.e., alternative interpretations of the data) leads to complex evidential/belief interrelationships among hypotheses. In the

[6] When we talk about "hypotheses" in this paper, we are really referring to what are called hypothesis *extensions* in RESUN/DRESUN (Carver & Lesser 1991a).

[7] DRESUN requires that hypotheses have sufficient attributes to be able to judge the consistency of new evidence without having to have access to all the existing evidence for the hypotheses. Thus, the consistency of two track hypotheses can be judged without requiring access to their supporting evidence.

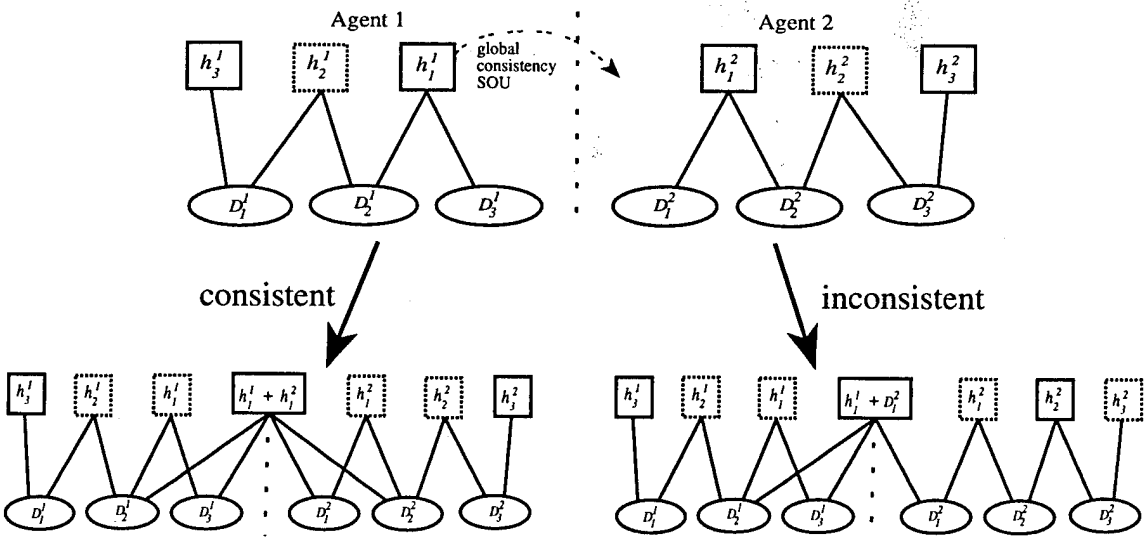


Figure 3: An example of the resolution of a global consistency SOU.
When there is a consistent explanation in the external agent, resolution of the global SOU associated with h_1^1 results in the creation of a merged hypothesis as a new alternative explanation in each agent. When the local hypothesis is inconsistent with hypotheses in the external agent, new alternatives may be created (as shown here). When the local hypothesis is inconsistent with the data in the external agent, new evidential links are created to represent the contradictory evidence.

case of only local evidence, these relationships can at least be properly evaluated. For example, in the inconsistent local evidence example in Figure 4, suppose that T_1 and $T_{2'}$ overlap at V_3 and V_4, but are inconsistent. The inconsistency is recognized because T_1 and $T_{2'}$ are alternative explanations for the shared V_3 and V_4 support, which allows the negative evidential relationship between T_1 and $T_{2'}$ to be evaluated properly.

Now, consider the case in which T_1 and $T_{2'}$ are inconsistent, but $T_{2'}$ is an external hypothesis. It is still straightforward to detect the inconsistency. However, because the local agent does not have any of $T_{2'}$'s supporting evidence, this inconsistency can be represented only as negative evidence for T_1—which again makes it impossible to precisely evaluate the belief in T_1. First, the effect that alternative interpretations have on each other's belief depends on the relative belief of the shared and non-shared portions. For instance, if the belief in $T_{2'}$ is largely due to the quality of the overlap data, then $T_{2'}$ does not represent strong belief against T_1. Second, evidential propagation now does not automatically reflect correct beliefs if there are other interrelated hypotheses. Suppose, for example, that there are additional local hypotheses that are inconsistent with T_1 (because they are alternative explanations for some of T_1's support). These hypotheses may also be inconsistent with the external $T_{2'}$, or they may be consistent with it. Unless these relationships are explicitly examined, this will lead to additional uncertainty in the belief in T_1.

This brief example shows that communication of abstract, incomplete hypothesis information can lead to uncertainty about the effect of external evidence on local hypotheses. Communicating incomplete information can still be useful, however. In some situations the uncertainty may not be critical to resolve, and if it is, it may be able to be used to guide further communcation. We have addressed this and a number of other representation issues in our extensions to DRESUN. Several of these extensions will be described in the example in the next section. As part of our extension of the model of external evidence, we have given DRESUN agents the ability to:

- link multiple views of a hypothesis based on external evidence at different levels of abstraction;
- link multiple hypothesis extensions that are being used in alternative local interpretations (e.g., different portions of a single external vehicle track hypothesis);
- locally create alternative versions of external hypotheses based on incomplete information and represent the resulting uncertainties;
- reformulate hypotheses for more efficient exploration and representation of alternatives;
- communicate back results of integrating information that was sent by another agent;
- avoid circular reasoning when exchanging evidence among agents;
- identify when shared information should be updated.

Resolving Global Inconsistency: An Example

In this section, we will return to the example of Figure 1. Our purpose will be to show how DRESUN's

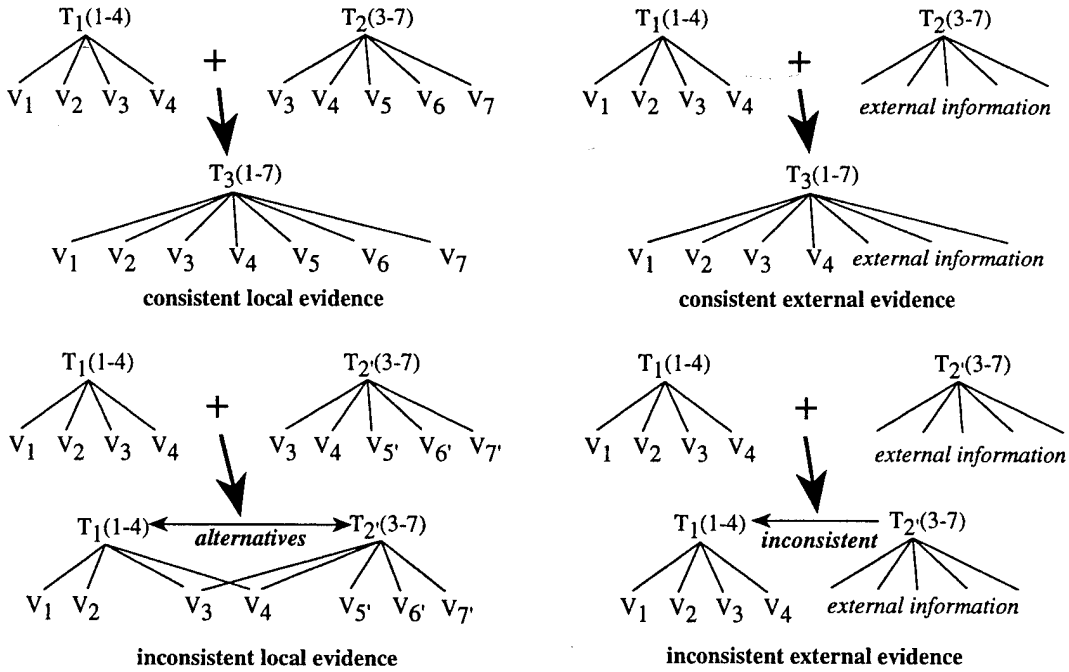


Figure 4: Examples of the differences in representing local and external evidence.

extended representation of external evidence provides the flexibility to be very efficient about the information that must be communicated among agents to resolve global inconsistencies. For the purpose of our presentation, we will assume that agent A and agent B have already formed their local track hypotheses (T_a and T_b) that extend through the overlap region. This results in *consistent-overlapping-model* SOUs being posted in each agent's PS-Model. We also will assume that these SOUs are not pursued until some level of confidence in the track hypotheses is reached (based on the local evidence), and that agent A is the first to communicate about its SOU.

When agent A starts initiates a dialog (with agent B) to resolve its "overlap" SOU, there are two options depending on whether agent A thinks the bulk of the processing to check consistency should be done by itself or by agent B: it could request agent B to send its best interpretations that cover the overlap region and then check consistency itself or it could send track T_a to agent B and let that agent check consistency. Likewise, if agent A chooses to send T_a it has several options in terms of the amount of detail it sends about T_a, or if agent B is requested to send back its interpretations it has several representation options. Here, we make the assumption that agent A will handle consistency checking and that potential solutions will initially be communicated at their most abstract level: sending only the attributes and degree of belief in the most likely top-level, track hypotheses.

Given these decisions, agent A requests that agent B send it any relevant potential solutions and agent B responds with track T_b. Agent A finds that track T_b is inconsistent with its own track T_a since the tracks overlap but cannot be merged. Because T_b is inconsistent with T_a, *negative external evidence* is created for T_a. This is the second stage of agent A's representation shown in Figure 5. The creation of this negative external evidence will cause a *global-inconsistency* SOU to be added to agent A's PS-Model. Whether or not this "inconsistency" SOU results in further communication or other processing depends on several factors, including: the original belief in T_a, the uncertainty about the magnitude of the (negative) effect that T_b has on T_a due to incomplete information about the external hypothesis (as described in the Representing External Evidence Section), the ability of agent A to pursue other sources of uncertainty in T_a (to locally resolve the uncertainty in T_a), the general classes of uncertainty affecting other hypotheses, the global consistency termination criteria, and so on.

Assuming that the agent chooses to pursue the "inconsistency" SOU, it first identifies plans that are relevant to resolving the SOU. One plan that we have developed for resolving a *global-inconsistency* SOU is applicable only when the inconsistency involves track hypotheses that are *partially consistent*. This plan attempts to construct a new extension of the local track that is consistent with a portion of the external evidence. Exactly which of the possible alternatives it initially chooses to create depends on the information it has about the relative credibility of the various por-

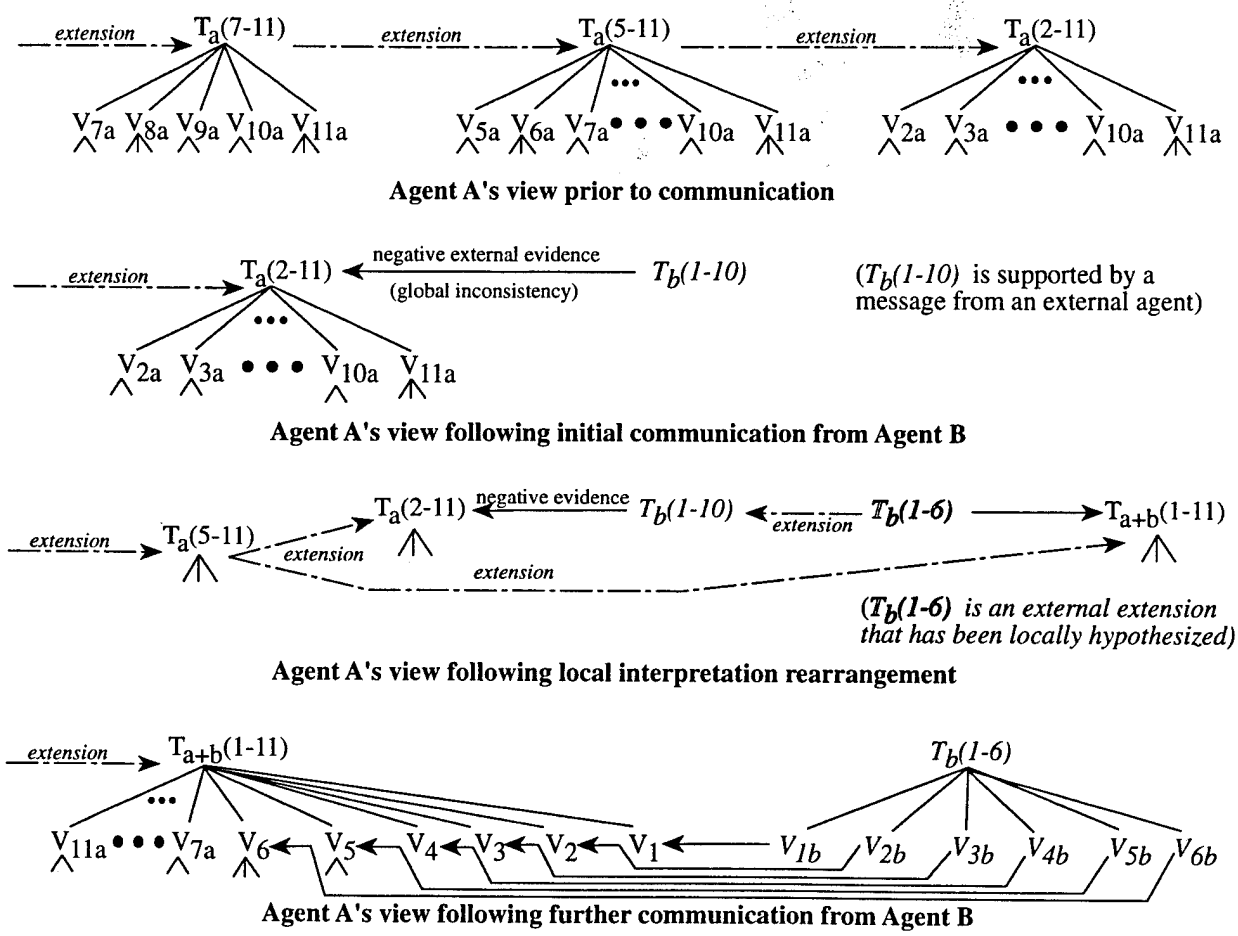


Figure 5: Agent A's evolving representation of evidence for the example.

tions of the inconsistent tracks. Here, agent A knows about the credibility of portions of only its own track T_a: the support from times 5 through 11 is quite strong and that from times 2 through 4 is weak (see Figure 1). Starting with the consistent portion of T_a at times 5 and 6, and the better supported portion from times 7 through 11, agent A decides to create a new track extension using local evidence from times 5 through 11.

In the third stage of Figure 5, agent A has created track T_{a+b} based on $T_a(5-11)$ (one of the intermediate extensions of T_a) and $T_b(1-6)$. $T_b(1-6)$ is an intermediate extension of the external track T_b that has been *locally hypothesized* by agent A—i.e., agent A has created this extension without knowing whether agent B has a representation of this version of T_b and without knowing the degree of belief in this portion of T_b.[8] Because agent A lacks both the supporting evidence and the overall belief rating for $T_b(1-6)$, there is considerable uncertainty about the belief rating for T_{a+b} (remember, it has only the overall rating for $T_b(1-10)$). The reasons for its uncertainty are represented by an SOU that is posted with T_{a+b}.

Assuming that agent A decides to pursue T_{a+b} further, since it is a credible globally consistent solution, this SOU drives the selection of a plan that requests agent B to communicate the support belief summaries for $T_b(1-6)$.[9] When the requested information is received, it is integrated into agent A's incomplete representations of $T_b(1-6)$ and T_{a+b}. The result of this process is shown in the final stage of Figure 5. With this level of information, agent A can evaluate the likelihood of T_{a+b} (given the evidence gathered so far by itself and agent B). Depending on the results of this evaluation and the termination criteria, agent A may then consider T_{a+b} to be a (likely) solution or to not be a solution, or it may need to try to gather additional

[8] Agent B may or may not have created this particular extension depending on how it gathered the evidence to construct $T_b(1-10)$. Agent B may have to reformulate its view of T_b in order to create $T_b(1-6)$.

[9] This information is provided in our application by the belief ratings at the vehicle (position) level and in the *uncertain-support* SOUs at the track level.

evidence to resolve the remaining uncertainty.

This example shows how DRESUN agents can carry on dialogs in order to resolve global inconsistencies. It also shows that these dialogs can be directed, using information at appropriate levels of detail, in order to limit the amount of information that must be communicated (and integrated). In this example, agent A does not have to have complete knowledge of T_b's supporting evidential structure or its raw data. All that is needed is information about the "quality" of the data sets supporting T_b. In fact, because agent A was able to to construct alternative hypotheses based on its local data and an incomplete view of T_b, it was able to limit the information it required to just a portion of T_b's support. The flexibility to do this sort of local processing is possible because DRESUN agents represent the uncertainty that results from the use of incomplete external evidence.

Conclusions

In this paper, we have discussed some of the issues that can arise in a DSA system when sharing incomplete or inconsistent information, or information at different levels of detail. These are important issues since the sharing of partial results is a critical element of an FA/C approach to distributed problem solving. We have also shown how the agent architecture of the DRESUN testbed has been extended to give the agents the flexibility to communicate information in a very directed manner. These extensions have focused on representing the uncertainties that occur when DRESUN agents exchange incomplete information, determining precisely what information is needed to resolve global inconsistencies, and reformulating hypotheses to more efficiently pursue alternative interpretations.

Because DRESUN supports a range of methods for resolving interpretation uncertainty and global inconsistency, coordination strategies must consider a variety of questions about whether/when/how to pursue interpretations and SOUs (the example in the previous section mentioned a number of options faced by the agents). We are pursuing both analytical and experimental approaches to determine appropriate coordination strategies (Decker & Lesser 1995), are analyzing the quality of solutions that can be produced by FA/C-based DSA systems (Carver & Lesser 1994), and are developing methods for analyzing the inherent complexity of interpretation scenarios (Whitehair & Lesser 1993). Since it is difficult to evaluate a framework independently of the strategies that are encoded within it, the development of suitable coordination strategies is a major focus of our current research.

References

Norman Carver and Victor Lesser, "A New Framework for Sensor Interpretation: Planning to Resolve Sources of Uncertainty," *Proceedings of AAAI-91*, 724–731, 1991.

Norman Carver, Zarko Cvetanovic, and Victor Lesser, "Sophisticated Cooperation in FA/C Distributed Problem Solving Systems," *Proceedings of AAAI-91*, 191–198, 1991.

Norman Carver and Victor Lesser, "A Planner for the Control of Problem Solving Systems," *IEEE Transactions on Systems, Man, and Cybernetics*, special issue on Planning, Scheduling and Control, vol. 23, no. 6, 1519–1536, 1993.

Norman Carver and Victor Lesser, "A First Step Toward the Formal Analysis of Solution Quality in FA/C Distributed Interpretation Systems," *Proceedings of the 13th International Workshop on Distributed Artificial Intelligence*, July, 1994.

Norman Carver, "Examining Some Assumptions of the FA/C Distributed Problem-Solving Paradigm," *Proceedings of the Midwest Artificial Intelligence and Cognitive Science Society Conference*, April, 1995.

Gregory Courand, "Cooperation via Consensus Formation," *Proceedings of the 10th International Workshop on Distributed Artificial Intelligence*, 1990.

Keith Decker, and Victor Lesser, "Designing a Family of Coordination Algorithms," *Proceedings of the International Conference on Multiagent Systems*, June, 1995.

Edmund Durfee and Victor Lesser, "Partial Global Planning: A Coordination Framework for Distributed Hypothesis Formation," *IEEE Transactions on Systems, Man, and Cybernetics* vol. 21, no. 5, 1167–1183, 1991.

Michael Huhns and David Bridgeland, "Multiagent Truth Maintenance," *IEEE Transactions on Systems, Man, and Cybernetics*, special issue on Distributed Artificial Intelligence, vol. 21, no. 6, 1437–1445, 1991.

Victor Lesser and Daniel Corkill, "Functionally Accurate, Cooperative Distributed Systems," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 11, no. 1, 81–96, 1981.

Victor Lesser and Daniel Corkill, "The Distributed Vehicle Monitoring Testbed: A Tool for Investigating Distributed Problem Solving Networks," *AI Magazine*, vol. 4, no. 3, 15–33, 1983 (also in *Blackboard Systems*, Robert Engelmore and Tony Morgan, editors, Addison-Wesley, 1988).

Victor Lesser, "A Retrospective View of FA/C Distributed Problem Solving," *IEEE Transactions on Systems, Man, and Cybernetics*, special issue on Distributed Artificial Intelligence, vol. 21, no. 6, 1347–1362, 1991.

Robert Whitehair and Victor Lesser, *A Framework for the Analysis of Sophisticated Control*, Technical Report 93-53, Computer Science Department, University of Massachusetts, 1993.

APPENDIX B

An Example of the Integration of Planning and Scheduling *

Alan Garvey and Victor Lesser

Computer Science Department
Lederle Graduate Research Center A243
University of Massachusetts
Amherst, MA 01003

Phone: (413) 545-3444
Fax: (413) 545-1249
Email: {garvey,lesser}@cs.umass.edu

September 13, 1995

Abstract

When solving real-time AI problems, it is often useful to have multiple methods available for solving subproblems. This allows acceptable solutions to be found even in tight deadline situations. However, it requires that problem solvers have both a simple planning component (to decide which methods to use) and a scheduling component (to schedule the execution of the chosen methods). In this paper we describe the design-to-time approach to real-time problem solving that uses this integration of planning and scheduling to find solutions to hard real-time problems. We describe the design-to-time algorithm and provide some experimental results that show the usefulness of this algorithm.

*This material is based upon work supported by the National Science Foundation under Grant No. IRI-9208920, NSF contract CDA 8922572, DARPA under ONR contract N00014-92-J-1698 and ONR contract N00014-92-J-1450. The content of the information does not necessarily reflect the position or the policy of the Government and no official endorsement should be inferred.

1 Introduction

In our work on real-time artificial intelligence (AI) we have found it useful to integrate a simple form of planning with scheduling to find good solutions. Planning is involved because multiple (often approximate) methods are available for many subproblems and it is necessary to decide which methods to execute. Having multiple methods available allows solutions to be found for a broad range of environmental situations. Scheduling is involved because there are hard deadlines by which subproblems must be completed, so it is necessary to understand when particular methods will complete execution and what results can be expected.

Steve Smith has talked about the usefulness of integrating planning and scheduling in complex, resource-constrained problems [9]. He discusses the need to combine deciding "what" to do (i.e., finding ways to satisfy goals from more basic descriptions of how domain actions affect the world) with "when" to do it (i.e., allocating resources over time to tasks in a way that maximizes overall system performance). While our approach is quite different from Smith's, the basic idea of integrating planning and scheduling to solve problems is shared.

Our approach to real-time problem solving is known as design-to-time [5, 6] and integrates planning and scheduling to solve real-time problems where multiple methods are available for many subproblems. Our integration differs from Smith's in that our planning and scheduling both address different kinds of problems in different ways. Our simple planner has to decide which of many combinations of solution methods to use to solve each problem, rather than determining from first principles what operators could be applied in each situation. Unlike Smith, our scheduler does not schedule resources (other than a CPU) and it schedules a single line of control. This differentiates our work from much of the standard AI scheduling work [8, 3, 7], although (as discussed at the end of the paper) we have begun to extend our scheduling mechanism to handle multiple lines of control and other resources. Figure 1 is a high level depiction of the design-to-time algorithm.

In design-to-time we frame the input problem in a generic representation of task structures known as TÆMS [2, 1]. In TÆMS we can describe problems in terms of how quality is incrementally accumulated over time, what soft and hard interactions there are between tasks, and how much uncertainty there is in method quality and duration. Figure 2 shows an example of a simple TÆMS task structure, in this case one that describes the problem of making coffee. Note that there are multiple methods for solving each of the three subtasks of the main problem.

The TÆMS framework gives us what might be called a partially digested description of a problem. We do not have to plan completely from scratch, deciding what operators apply in a given situation and how they can be combined to solve problems. That information is available to us in the TÆMS task structure. What we need to do is determine which of the many options available in that task structure we should actually pursue. Because we are dynamically combining options together (rather than using predefined process plans) TÆMS needs to explicitly represent the soft interactions

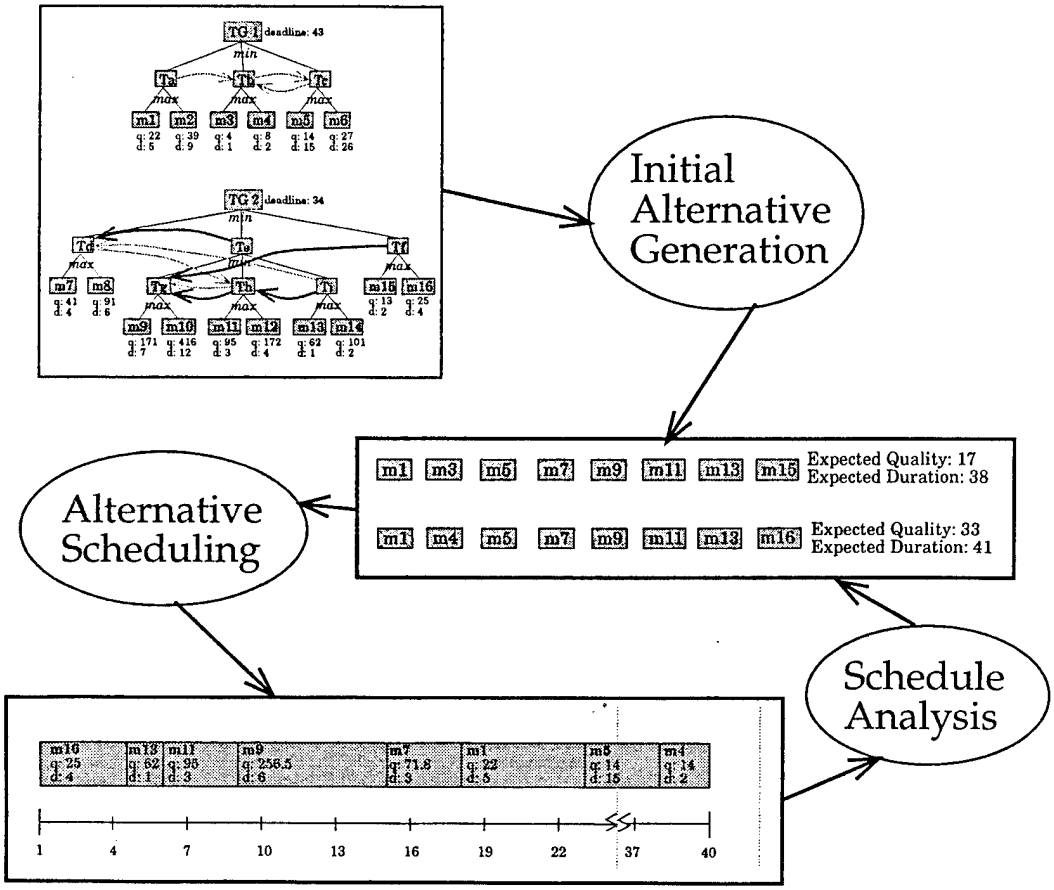


Figure 1: A heuristic algorithm for integrating planning and scheduling.

between tasks that could normally be built in to the process plan for a task, because the exact effect of the interactions depends on the context in which the tasks are executed.

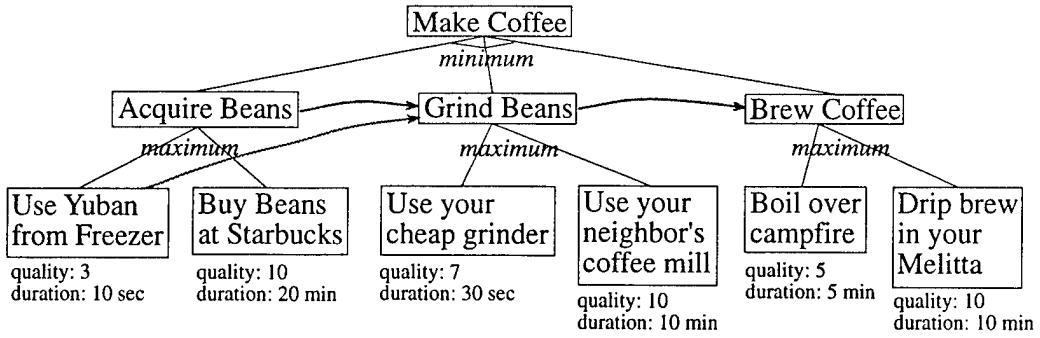


Figure 2: An example of TÆMS task structure to be scheduled. The black lines represent task/subtask connections, the thin gray line represents a facilitates relationship and the thick gray lines represent enables relationships.

In a TÆMS task structure the leaves of the graph represent executable computations (known as *methods*) and the nonleaf nodes represent tasks that achieve quality as a function of the quality of their subtasks. Each separate graph is known as a *task group* and represents a single independent problem to be solved. Each task group has a deadline by which all computation on that task group must be completed. Non-parent-child connections between tasks and methods represent interactions, such as *enables* (Task A must have quality greater than a threshold before Method B can correctly begin execution) and *facilitates* (if Task A has quality greater than a threshold, then Method B will have reduced duration and/or increased quality).

The goal of our design-to-time scheduling algorithm is to maximize the overall quality achieved. What quality means depends on the particular application, but components often include completeness (a more complete answer usually has higher value), certainty (increasing the likelihood that a result is correct can increase its quality), accuracy (a more accurate answer often has higher quality) and cost (the quality of a solution could, for example, be inversely related to the cost, where cost is a measure of non-CPU resources required for a solution).

In the coffee example from Figure 2 overall quality for the task group is the minimum of the qualities achieved by acquiring beans, grinding and brewing. Acquiring beans enables grinding. You have to have the beans before you can grind them. Grinding enables brewing. You can't actually brew coffee until you have ground the beans. Each of these intermediate tasks has two different methods for achieving it that have different expected quality and duration values. Using Yuban from the freezer reduces the expected duration of the grinding task (because presumably the Yuban is already ground, although it might be desirable to grind it more finely). This translates to a *facilitates* in TÆMS.

Another issue that can arise in problems of this form is uncertainty. Often we might only approximately know how long a particular solution method will take or

what quality of solution to expect. Maybe there will be a big line at Starbucks today (increased duration) or maybe they will be out of our favorite variety and we will have to settle for something else (reduced quality).

Given an input problem of this form, the design-to-time algorithm first generates a set of *alternatives*, which are unordered combinations of executable methods that it thinks might be good for solving the problem. These alternatives are chosen without taking interactions and deadlines into account (because taking them into account requires actual scheduling, which is the next step of the algorithm). These alternatives are generated using a dynamic programming technique and represent a good cross-section of possible solutions with different resource (in this case CPU) requirements. This is the planning part of the algorithm that is choosing which of many possible combinations of methods are worth considering in detail.

Some of the alternatives that would be generated for the coffee making example include:

Yuban-grinder-boil expected quality: 3 (maximum of 3, 7, 5), expected duration: 5 min. 40 sec.

Starbucks-mill-drip expected quality: 10, expected duration: 40 min.

Starbucks-grind-boil expected quality: 5, expected duration: 25 min. 30 sec.

Starbucks-grind-drip expected quality: 7, expected duration: 30 min. 30 sec.

After the initial set of alternatives is generated the algorithm goes into a loop of choosing an alternative to schedule, scheduling the chosen alternative and evaluating the resulting schedule, which may result in new alternatives being generated. Choosing an alternative to schedule is done in a heuristic manner that takes into account deadlines, the best schedules found so far and the expected results of the alternatives. A schedule is produced for an alternative using a greedy heuristic approach taking into account factors such as expected quality and duration, whether a method interacts with other methods, and deadline. Evaluation includes both checking the new schedule to see if it is better than the best schedule currently available (and if it is remembering it), as well as looking for ways to improve the schedule. For example, if a method in the alternative could not be scheduled because it was not enabled, a new alternative will be generated that includes all of the methods in the scheduled alternative plus a method or methods to enable the unenabled method.

In the simple coffee example it is possible to enumerate all reasonable schedules and optimally pick the best one (which, of course, depends on the deadline). However, in general, when there are m methods, there are 2^m possible combinations of these methods (the power set of m). Each of these combinations has more than $i!$ possible orderings where i is the number of methods in a particular combination. A problem with that kind of exponential growth clearly requires a heuristic solution technique.

In the remainder of this paper we first describe our design-to-time scheduling algorithm, including its mechanisms for handling task interactions and uncertainties.

Then we describe the results of experiments run to understand under what environmental conditions the design-to-time approach could be useful. Finally we draw some conclusions and discuss future work.

2 Heuristic Design-to-time Scheduling Algorithm

Our heuristic design-to-time scheduling algorithm consists of three main components: a simple planner that generates alternatives (unordered combinations of executable methods), a scheduler that assigns start and finish times to the methods from an alternative, and an evaluator that both remembers the best schedules found so far and looks for ways to improve the current schedule. Figure 1 provides a graphical picture of the connected components of this algorithm.

That the algorithm consists of these three particular components is because they help to reduce the overall complexity of the problem to manageable levels. Initial planning is done at a higher level of abstraction, ignoring interactions between tasks and deadlines, to allow us to feasibly consider a broad cross-section of possible solution plans. Scheduling is done on only those plans that make it through the first level of the algorithm, because completely scheduling all possible plans would be computationally impossible in most situations. Evaluation is done at the end to reduce the likelihood that we are missing good plans by suggesting plans that are simple variations on existing plans that could have increased value.

The input to the algorithm is a TÆMS task structure that describes the set of problems to be solved, the interactions within those problems, uncertainties about the expected quality and duration of methods, and the deadlines by which problems must be solved. An example of such as task structure is shown in Figure 2.

The first step of the algorithm is to generate a set of alternatives for solving the overall problem. This done by recursively generating alternatives from the leaves of the TÆMS task structure up to the root. For a standard method there is just one alternative, containing the method, with the expected quality and duration of the method. Our algorithm also supports anytime methods (i.e., methods that are anytime algorithms that always have a result at hand and return higher quality results as they are given more execution time, up to some bound). For anytime methods we generate a small fixed number of alternatives by choosing points on the *performance profile* (a function mapping duration to expected quality) of the anytime method. These points are chosen in a way that minimizes the difference in estimated quality and duration between the chosen points.

For higher level tasks in the TÆMS task structure alternatives are generated from the alternatives for subtasks, depending on the quality accumulation function of the task. The goal is to generate plans with different resource requirements to allow planning at the next level to have a good cross-section of choices. This is done using a dynamic programming approach that finds the highest expected quality combination of alternatives for subtasks, for each possible duration for the task. Note that the

estimated duration and quality associated with an alternative are often not accurate because they do not take interactions between tasks into account. To fully take these interactions into account requires scheduling, which we don't do at this stage of the algorithm, because it would be much too computationally expensive. The resulting list of alternatives for a task is pruned to a manageable number and becomes the input for generating the alternatives for its supertask.

Once a set of alternatives has been generated for the overall set of problems to be solved, the next set of steps in the algorithm begin. This is a loop that first chooses an alternative to schedule, then schedules that alternative, then evaluates the resulting schedule possibly generating additional alternatives. This loop continues until either no known alternative could produce a better schedule that the best current schedule or all alternatives are exhausted or a threshold number of alternatives has been scheduled.

The next alternative to schedule is chosen using a heuristic algorithm with the following priorities:

1. If no schedules have been found, choose the fastest alternative that would achieve quality for all problems.

2. If the best schedule found so far achieves a total quality of q and has total duration d, choose alternatives that could achieve quality greater than q or could achieve at least quality q in duration less than d.

Dynamic duration thresholds are used to limit the set of alternatives considered to those with an estimated duration below a threshold. This threshold is set depending on the estimated duration of the alternative for the best schedule found so far, on how long we have been scheduling since finding a new schedule, and a few other considerations.

The next step is to actually schedule the chosen alternative. The algorithm for doing this is very simple. Remember that an alternative is just an unordered set of methods. Scheduling is done by rating each method using a set of heuristics. The highest rated method is added to the schedule, then the remaining methods are rated again. This continues until all methods are scheduled or no method is rated as acceptable for adding to the schedule. Heuristics include preferring methods with earlier deadlines, rating as unacceptable any method that is not enabled, and preferring methods that enable other methods.

Once a schedule has been produced, it is evaluated. If it is better than the best current schedule, it becomes the best schedule. The evaluation criteria include how many problems are solved, the total quality of the solutions (higher quality preferred), the total duration of the schedule (shorter duration preferred), and the likelihood that the schedule will perform as expected (higher certainty preferred).

Schedules are also looked at by various *improvement* operators that try to find ways to improve the schedule. For example, one improvement notices when methods in an alternative could not be scheduled because they were not enabled and adds new methods to the alternative to enable those methods. This new alternative is added to the list of alternatives that is considered for scheduling. Once a schedule has been

considered for improvements, we go back to the beginning of the loop, choose another alternative to schedule and continue.

3 Experimental Results

These experiments have two main goals. One is to show the usefulness of the integration of planning and scheduling. The other is to examine whether the environmental parameters that we consider most important (interactions between tasks and the availability of multiple methods) in fact can lead to improved system performance.

One of the useful features of the TÆMS task structure environment is that it is possible to generate many random problems for experimentation. TÆMS also supports easy replication of environments with just those parameters under study being varied, allowing us to do *matched pair* experiments. This means that, for example, when we vary the likelihood of facilitates, only the number of facilitates changes, all other aspects of the task structure remain the same for a set of runs. In the experiments described here 100 different environments were generated, then the parameters of interest were varied within those environments.

The first experiment looks at the importance of the planning component of our scheduling algorithm in the overall performance of the algorithm. In this experiment we examine the performance of our scheduling algorithm using our standard planning component compared to the performance after replacing that component with a random alternative generator and with a generator that generates one alternative containing all methods. The random alternative generator generates alternatives of a fixed size by randomly choosing methods. The expectation is that use of the planner will significantly improve overall system performance. Figure 3 shows proportion of expected quality versus deadline parameter for the three algorithm variations studied. As the deadline parameter gets higher, deadlines get looser allowing more time to be spent executing tasks. As this graph shows, the scheduler with the standard planning component outperforms the other two schedulers. A two-way analysis of variance (ANOVA) shows significant main effects for both the deadline parameter ($F = 194.5$, $p < 0.00001$) and which alternative generator is used ($F = 95.0$, $p < 0.00001$), as well as an interaction effect between the two ($F = 9.9$, $p < 0.00001$). Our interpretation of the interaction effect is that as deadlines become looser, the value of the planning mechanism becomes more important, because it allows facilitates to be fully taken advantage of. The overall result of this experiment is that the planning component leads to significantly better performance under all situations studied, and that the effect on performance becomes more positive as deadlines become looser.

The remaining experiments look at the effect of different environmental parameters on the performance of the system. In the first experiment we varied the number of leaf methods created at the points in the task structure generation process where methods are created. As additional methods are added they are given expected quality and duration as a proportion of the quality and duration of the first method created. That

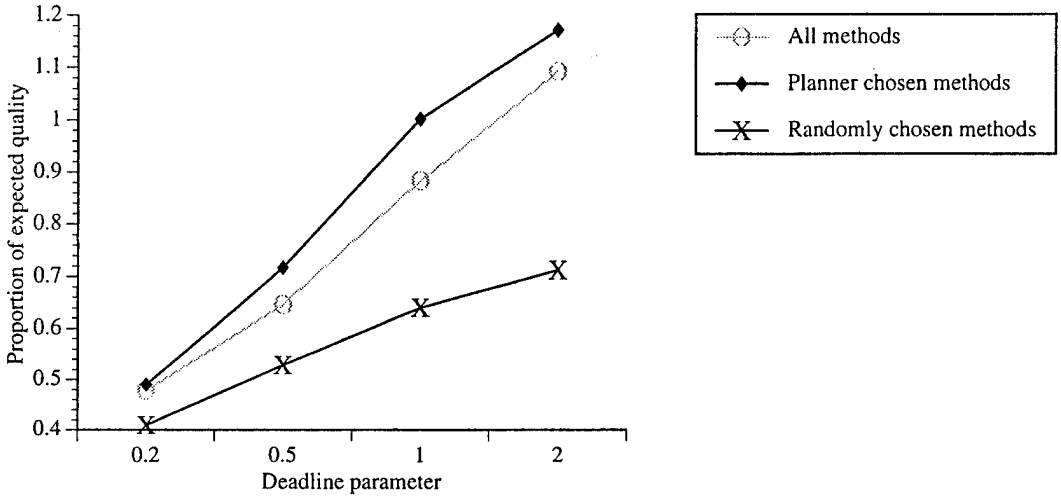


Figure 3: Deadline parameter setting versus mean proportion of expected quality for three scheduler configurations.

is, the first method has quality q and duration d. If there are two methods, the second one gets quality $q/2$ and duration $d/2$. If there are 5 methods, they get qualities of $q/5$, $2q/5$, $3q/5$, $4q/5$, and q with the corresponding durations. Figure 4 shows the effect of varying the number of leaf methods on the quality achieved. Quality is measured as a proportion of the expected quality (the maximum quality that could be achieved without deadlines or interactions between tasks). The expectation is that adding more leaf methods will result in more options, allowing higher quality to be achieved, especially at tighter deadline values. As figure 4 shows, this appears to be the case. Using the data from this experiment a two-way ANOVA shows a significant main effect for the number of leaf methods ($F = 11.8$, $p < 0.00001$). The effect of facilitates likelihood is not significant ($F = 1.66$, $p < 0.19$) and no interaction effect is detected. The overall result of this experiment is that adding additional methods (at least up to the total of 3 or 4) significantly improves system performance.

In the second experiment we looked at the effect of varying the likelihood of facilitates. As part of task structure generation, facilitates are added to the task structure depending on the likelihood of facilitates. In this experiment we varied facilitates likelihood from 0 (no facilitates) to 1 (facilitates at each place where one is possible). Figure 5 shows the proportion of expected quality (as in the previous experiment) versus the facilitates likelihood for four values of a deadline parameter at high facilitates strength. Facilitates strength refers to the power of the effect of the facilitates on quality and duration. The expectation is that more facilitates will allow higher quality to be achieved both because facilitates directly increase the quality of facilitated methods and because durations are reduced, possibly allowing more tasks to be completed. Using the data from this experiment we ran a two-way ANOVA. It showed a significant main effect of both deadline parameter ($F = 58.7$, $p < 0.00001$) and facilitates likelihood (F

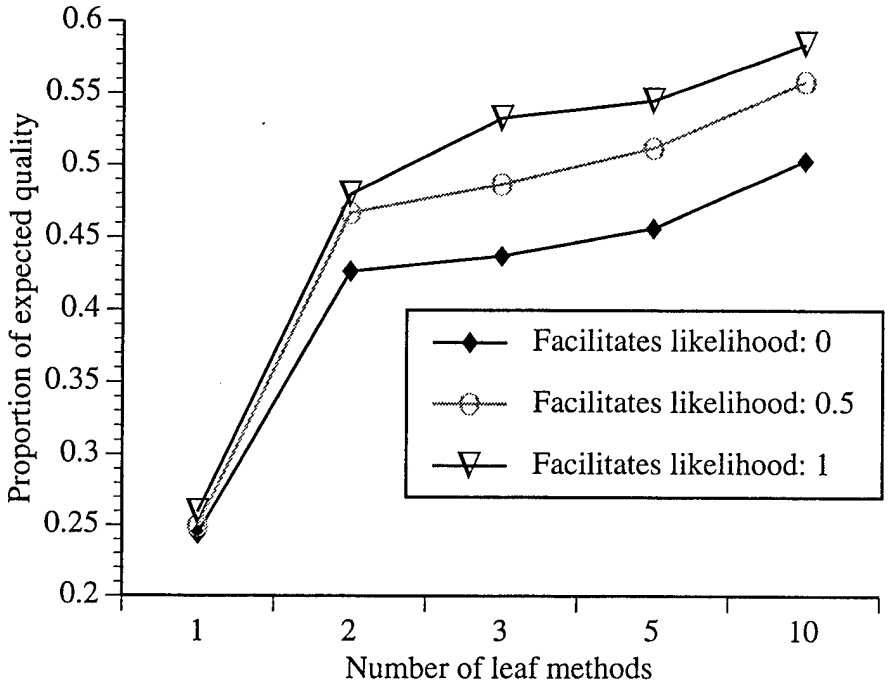


Figure 4: Number of leaf methods versus proportion of expected quality for three values of facilitates likelihood with tight deadlines

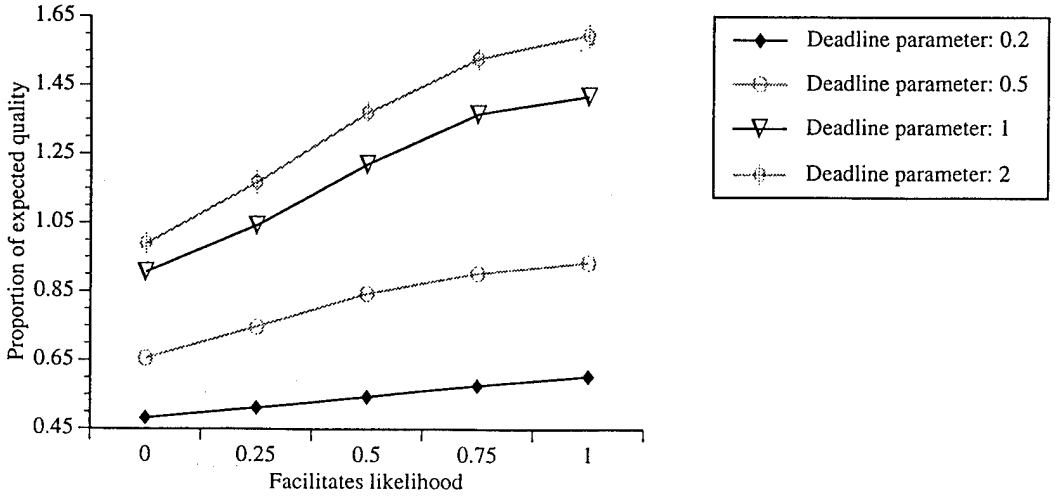


Figure 5: Facilitates likelihood versus proportion of expected quality for four values of a deadline parameter at high facilitates strength.

= 9.7, $p < 0.00001$). No significant interaction effect was detected.

One interesting question that can be asked about the result of this experiment is if the positive effect of the facilitates happens specifically because the scheduling algorithm takes facilitates into account or whether the effect would remain even if the algorithm did not pay attention to the facilitates. We examined this question by adding a parameter that allows the visibility of facilitates to the scheduler to be varied separate from the likelihood of facilitates appearing. Figure 6 shows the proportion of expected quality versus facilitates visibility for five values of facilitates likelihood at high facilitates strength. The facilitates visibility parameter affects the likelihood of a facilitates being detected by the scheduler, but does not affect its effect as methods are executed. The expectation is that as the likelihood that the scheduler sees facilitates increases, the quality increases, because the scheduler will be able to schedule to take advantage of facilitates (as opposed to taking advantage of them inadvertently). As figure 6

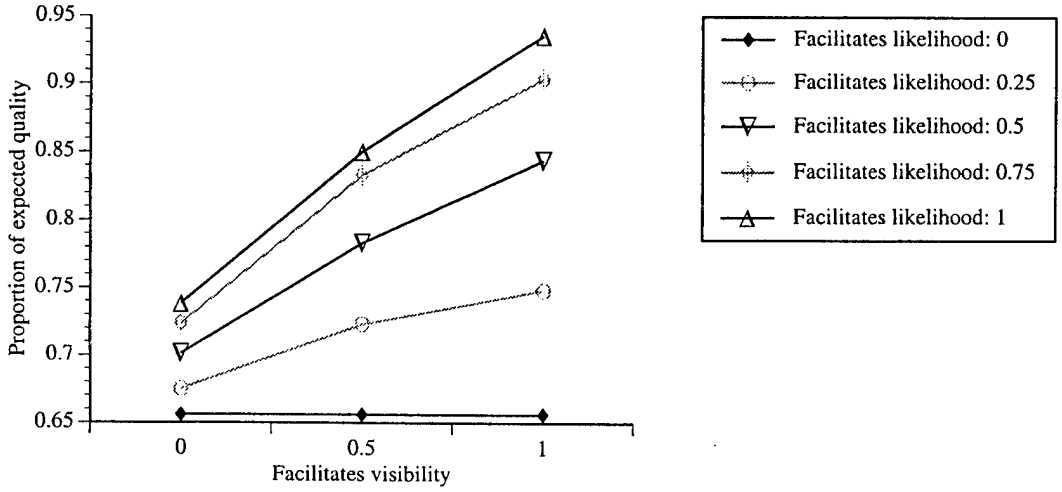


Figure 6: Likelihood of facilitates visibility versus mean proportion of expected quality for deadline parameter setting 0.5 and high facilitates power.

shows, quality increases significantly as facilitates visibility increases, particularly at higher facilitates likelihoods. This shows up in our two-way ANOVA test, which shows significant main effects for both facilitates visibility ($F = 18.9$, $p < 0.00001$) and facilitates likelihood ($F = 7.0$, $p < 0.00001$), as well as nearly showing an interaction effect between the two ($F = 1.79$, $p < 0.075$).

4 Conclusions

In solving real-time AI problems it can be useful to combine a simple planning mechanism with a scheduler to find problem solutions. Planning is useful for choosing which particular methods to combine to solve the problem at hand depending on how those methods interact and how tight the deadlines are. Scheduling is necessary to allow a

clear understanding of when and how particular subproblems will be solved. We have shown the design-to-time real-time scheduling algorithm for solving problems of this sort by combining a simple planning mechanism with a scheduler.

Through experimentation we demonstrated that several factors have a significant effect on the performance of the system. Having more methods available leads to increased quality. Having more facilitates to take advantage of also leads to increased quality. The quality increase is higher when facilitates are visible to the scheduler than when they are hidden from it but still present. Our simple planning mechanism outperforms the random choosing of methods and choosing all of the methods (forcing the scheduling component to choose which methods to schedule).

We have began to extend our approach to allow us to solve problems where non-CPU resources exist and multiple threads of control must be scheduled (e.g., parallel or distributed problem solving). TÆMS supports the representation of these resources, including resources that can be used by one thread of control at a time (such as machines in a job shop) and multiple access resources (such as a shared network whose performance degrades as its usage increases). Such an extension would allow us to solve a broader class of time-constrained problems.

This paper describes only a portion of the work that has been done on the design-to-time approach to problem solving. Other areas that have been investigated include uncertainty in method quality and duration, monitoring of method execution, embedding a scheduler in a larger application, and a significantly larger set of experiments that study the usefulness of the approach. Further details on this work can be found in the first author's dissertation [4].

References

[1] Keith S. Decker. *Environment Centered Analysis and Design of Coordination Mechanisms.* PhD thesis, University of Massachusetts, 1995.

[2] Keith S. Decker and Victor R. Lesser. Quantitative modeling of complex computational task environments. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 217–224, Washington, July 1993.

[3] Mark S. Fox, Norman Sadeh, and Can Baykan. Constrained heuristic search. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 309–315, Detroit, Michigan, August 1989.

[4] Alan Garvey. Design-to-time real-time scheduling. Ph.D. Dissertation (forthcoming), Department of Computer Science, University of Massachusetts, Amherst, MA, 1995.

[5] Alan Garvey, Marty Humphrey, and Victor Lesser. Task interdependencies in design-to-time real-time scheduling. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 580–585, Washington, D.C., July 1993.

[6] Alan Garvey and Victor Lesser. Design-to-time real-time scheduling. *IEEE Transactions on Systems, Man and Cybernetics*, 23(6):1491–1502, 1993.

[7] David W. Hildum. *Flexibility in a Knowledge-Based System for Solving Dynamic Resource-Constrained Scheduling Problems.* PhD thesis, Department of Computer Science, University of Massachusetts, Amherst, September 1994.

[8] Peng Si Ow and Stephen F. Smith. Viewing scheduling as an opportunistic problem-solving process. In *Annals of Operations Research 12*, pages 85–108. J.C. Baltzer AG, Scientific Publishing Company, 1988.

[9] Stephen F. Smith. Integrating planning and scheduling: Towards effective coordination in complex, resource-constrained domains. In *Keynote Address at the Italian Planning Workshop*, Rome, Italy, September 1993.

APPENDIX C

Analyzing the Need for Meta-Level Communication [1]

Keith S. Decker and Victor R. Lesser
Technical Report 93–22 (IJCAI-93 + AAAI-93)

UMass Computer Science Technical Report 93–22
May 1, 1995

Abstract

In naturally distributed, homogeneous, cooperative problem solving environments where well-defined tasks arrive at multiple locations, decisions must be made about the extent of, and overlap between, each agent's area of responsibility—the agents' *organization.* The organization may be constructed statically by a system designer, or dynamically by the agents during problem solving. No one organization is optimal across environments or even specific problem solving instances [6, 7, 8].

This paper presents an analysis of static and dynamic organizational structures for this class of environments, exemplified by distributed sensor networks. We first show how the performance of any static organization can be statistically described, and then show under what conditions dynamic organizations do better and worse than static ones. Finally, we show how the variance in the agents' performance leads to uncertainty about whether a dynamic organization will perform better than a static one given only agent *a priori* expectations. In these cases, we show when meta-level communication about the actual state of problem solving will be useful to agents in constructing a dynamic organizational structure that outperforms a static one. Viewed in its entirety, this paper also presents a methodology for answering questions about the design of distributed problem solving systems by analysis and simulation of the characteristics of a complex environment rather than by relying on single-instance examples.

[1] Portions of this technical report appeared in the proceedings of IJCAI-93 'An approach to analyzing the need for meta-level communication', and AAAI-93 'A one-shot dynamic coordination algorithm for distributed sensor networks'. This work was supported by DARPA contract N00014-92-J-1698, Office of Naval Research contract N00014-92-J-1450, and NSF contract CDA 8922572. The content of the information does not necessarily reflect the position or the policy of the Government and no official endorsement should be inferred.

1 Introduction

Organizational theorists have long held that the organization of a set of agents cannot be analyzed separately from the agents' task environment, that there is no single best organization for all environments, and that different organizations are not equally effective in a given environment [8]. Most of these theorists view the uncertainties present in the environment as a key characteristic, though they differ in the mechanisms that link environmental uncertainty to effective organization. In particular, the *transaction cost economics* approach [11] focuses on the *relative efficiencies* of various organizations given an uncertain environment, while the modern *contingency theory* approach [14] focuses on the need for an organization to expand toward the earliest available *information that resolves uncertainties* in the current environment.

In this paper we use both of these concepts to analyze potential organizational structures for a class of naturally distributed, homogeneous, cooperative problem solving environments where tasks arrive at multiple locations, exemplified by distributed sensor networks [10]. Previous approaches to analyzing organizations in distributed sensor networks have either not focused on the effectiveness of the organization [2, 12], or have only analyzed organizational effectiveness in particular, single-instance examples [7]. Our approach is to model the task environment mathematically, using a formalism developed specifically to study distributed coordination and scheduling [5]. We then develop expressions for the *expected efficiencies* of static and dynamic organizational structures, in terms of the cost of communication and time to complete a given set of tasks. Finally, we validate these mathematical models by using simulations.

A dynamic organization is one in which the responsibilities of agents can be reassigned based on a developing view of the problem at hand. Due to the uncertainties explicitly represented in the task environment model, there may not be a clear performance tradeoff between static and dynamic organizational structures. Agents that have a dynamic organization have the option of meta-level communication—communicating about the current state of problem solving as opposed to communicating about solving the problem itself. In this way, *information that resolves uncertainties* about the current environment becomes available to the agents, allowing the agents to then create the most efficient organization for the situation.

Section 2 describes the task environment model, the assumptions behind it, and analyzes the uncertainties present. Section 4 describes static and dynamic organizational structures, and develops expressions for the expected performance of each organizational style. This section also describes a particular algorithm for generating dynamic structures, and discusses the actual performance of the algorithm. In Section 5 we then show how the variance in performance without communication can lead to the efficient use of meta-level communication to customize a dynamic organizational structure. Finally, we discuss how these results can be used by designers of distributed problem solvers, and how our methodology can be used by other researchers. Throughout each section, we will illustrate and confirm the analytical results experimentally, using as an example a simulated distributed sensor network similar to the Distributed Vehicle Monitoring Testbed (DVMT) [10].

2 Task Environment Model

Our task environment model of naturally distributed problems assumes that several independent groups of tasks arrive at multiple locations over a period of time called an *episode*. For example, in a distributed sensor network (DSN) episode the movements of several independent vehicles will be detected over a period of time by one or more distinct sensors, where each sensor is associated with an agent. The performance of agents in such an environment will be based on how long it takes them to process all the task groups, which will include the cost of communicating data, task results, and meta-level communication, if any. The organizational structure of the agents will imply which subsets of

which task groups are available to which agents and at what cost. For example, if DSN agents have overlapping sensors, either agent can potentially work on data in the overlapping area (from its own sensor) without any extra communication costs. We make several simplifying assumptions: that the agents are homogeneous (have the same capabilities with respect to receiving data, communicating, and processing tasks), that the agents are cooperative (interested in maximizing the system performance over maximizing their individual performance), that the data for each episode is available simultaneously to all agents as specified by their initial organization, and that there are only structural (precedence) constraints within the subtasks of each task group.[1]

Any single episode can be specified by listing the task groups, and what part of each task group was available to which agents, given the organizational structure. Our analysis will be based on the statistical properties of episodes in an environment, not any single instance of an episode. The properties of the episodes in a DSN environment are summarized by the tuple $\mathcal{D} =< A, \eta, r, o, \mathcal{T} >$ where A specifies the number of agents, η the expected number of task groups, r and o specify the structural portion of the organization by the *range* of each agent and the *overlap* between agents, and $\mathcal{T}$ specifies the homogeneous task group structure (Section 2.5 and Figure 4 describes how task group structures are specified).

Our analysis initially focuses on exactly what *a priori* knowledge agents have about the distribution of task groups in an episode. First we will look at the distribution of the lowest-level sensor subtasks of a single task group among multiple agents (deriving the maximum expected number of subtasks), and then we will look at the distribution of task groups themselves. These results will then be used in subsequent sections to derive the total amount of work, and therefore expected termination performance, under various organizational structures and control schemes. Section 2.2 derives the expected number of low-level sensor subtasks at the most heavily-loaded agent given the number of task groups that same agent sees, how many agents see a single task group, and the environmental parameters. Section 2.3 then derives the number of task groups at the most heavily-loaded agent given the number of agents that see a single task group and the environmental parameters. Section 2.4 derives the number of agents that will see a single task group given the environmental parameters. Finally, Section 2.5 will show the structure of a task group, which allows us to derive the amount of work an agent must do given the number of sensor subtasks it executes, the number of task groups at the agent, and the number of agents involved in a single task group. All of these parts together allow us to derive the expected termination time of a particular system of agents.

2.1 Task environment simulation

In the next section and for the rest of the paper, we will test the model we are developing against simulated DSN problems. Each simulated DSN episode will take place on a grid where the concepts of length and size correspond directly to physical distances. For example, Figure 1 illustrates several simple organizations imposed on such a grid in our simulation.

In the simulation we assume that each vehicle is sensed at discrete integer locations (as in the DVMT), randomly entering on one edge and leaving on any other edge. Inbetween the vehicle travels along a *track* moving either horizontally, vertically, or diagonally each time unit using a simple DDA line-drawing algorithm (see Figure 5). In an 18×18 grid, the (empirical) average length of a track is 14 units—the actual length of any one track will range from 2 to 19 units and is not distributed normally. Given the organization (r, o, and A, and the geometry), we can calculate what locations are seen by the sensors of each agent. This information can then be used along with the locations traveled by each vehicle to determine what part of each task group is initially available to each agent. Section 2.5 will detail what the structure of each task group is for the DSN simulation.

[1] In general there are usually more complex interrelationships between subtasks that affect scheduling decisions, such as *facilitation* [4].

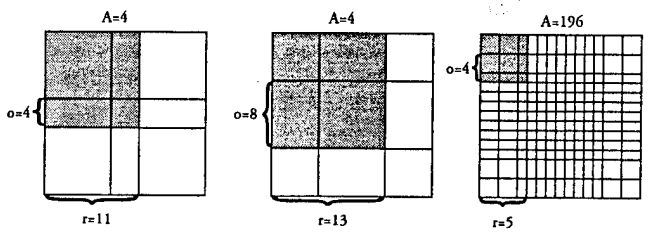


Figure 1: Examples of 18 × 18 DSN organizations

This specification is applicable to other interesting environments. For example: each task group may comprise several different types of subtasks; each agent may only respond to a certain type of subtask (its 'range'); multiple agents may respond to the same types ('overlap'). In general, the parameters of 'range' and 'overlap' can be multidimensional.

2.2 Expected number of sensor subtasks

In order to analyze the performance of a particular organization, we will want to know what proportion of each task group each agent is likely to process. There will be some upper limit on this proportion (related to the agent's range r), and sometimes the agent will process less than this upper limit. Especially in static organizational structures where tasks are not exchanged, the termination of the system as a whole can be tied to the completion of all tasks at the most heavily loaded agent. Normally, we would use the *average* part of a task group to be seen, but since the focus of our analysis is the termination of problem solving, we need to examine the *expected maximum* portion of a task group to be seen. This section will develop an equation for the expected maximum workload at an agent by counting the expected number of low-level sensor subtasks (each individually associated with a sensed vehicle location) that the maximally loaded agent will have.

The amount of a single task group seen by an agent (which is the same as the number of sensor subtasks in the DSN example) can be viewed as a random variable S with a probability density function and corresponding cumulative distribution function. In the DSN environment, S is discrete, and its probability function (determined empirically) is heavily weighted toward r (the maximum). To simplify the analysis, instead of letting S correspond to the number of subtasks in a single task group seen by an agent, let's create a new random variable $\mathbf{S}$ that we have equal 1 if the agent sees the maximum amount, and 0 otherwise. Now $\mathbf{S}$ has a Bernoulli (coin-tossing) distribution with parameter p corresponding to the chance of an agent seeing the maximum amount r of a task group. Let's assume we know that $N \leq n$ is the number of task groups at the maximally loaded agent, and that on average $a \leq A$ agents see a single task group (we'll remove these assumptions later). The number of times an agent sees the maximum out of N task groups (N coin flips) then has a binomial distribution ($b_{N,p}(s)$). We need to know, given that a agents each flip N coins, what the distribution is of the *maximum* number of 'heads' any agent sees—this is called the binomial max order statistic, $g_{a,N,p}(s)$:[2]

$$\begin{aligned} b_{N,p}(s) &= \binom{N}{s} p^s (1-p)^{N-s} && [\Pr[\mathbf{S} = s]] \\ B_{N,p}(s) &= \sum_{x=0}^{s} b_{N,p}(x) && [\Pr[\mathbf{S} \leq s]] \\ g_{a,N,p}(s) &= B_{N,p}(s)^a - B_{N,p}(s-1)^a && [\Pr[\hat{\mathbf{S}} = s]] \end{aligned}$$

The random variable $\mathbf{S}$ referred to *any* agent, the new random variable $\hat{\mathbf{S}}$ refers to the *maximally loaded agent*. It's probability function $g(s)$ has a much steeper shape and larger expected value than the binomial $b(s)$ (try it and see). In the DSN example we have $p = 0.5$[3] and the amount of a task group

[2] A more detailed derivation of this result is in our tech report.

[3] Empirically determined through simulation.

seen when an agent does not see the maximum amount averages $r/2$. Now we can convert back from the easy to analyze random variable $\hat{\mathbf{S}}$ to the variable we are really interested in, $\hat{S}$, by the equation $\hat{S} = (r\hat{\mathbf{S}} + (r/2)(N - \hat{\mathbf{S}}))$. The expected heaviest load seen by any agent when a agents see N task groups with a probability p of seeing r and probability $1 - p$ of seeing $r/2$ is:

$$E[\hat{S}|N, a] = \sum_{s=0}^{N} g_{a,N,p}(s)(rs + \frac{r}{2}(N - s)) \tag{1}$$

To reiterate: out of N task groups the maximally loaded agent sees the maximum r some $\hat{\mathbf{S}}$ (a random variable) times, and the other $(N - \hat{\mathbf{S}})$ times it sees only $r/2$. Eq. 1 shows the expected value of a new random variable $\hat{S}$ that indicates the number of sensor subtasks at the *maximally* loaded agent when there are N task groups in an episode. N is itself a random variable, we'll look at its distribution in the next section.

Figure 2 shows the heaviest load actually observed and averaged over 1000 runs, plotted against the expected value, for n tracks and all square DSN organizations [$2 \leq r \leq 10, 0 \leq o \leq r, 1 \leq \sqrt{A} \leq 10, 1 \leq N \leq 10$] ($R^2 > .98$)[4]. The colors of the points refer to the value of r; lighter grey corresponds to larger values of r.

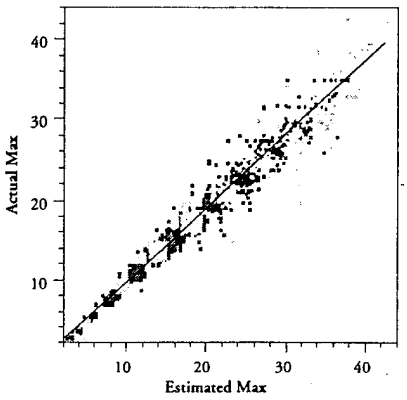


Figure 2: Actual versus predicted heaviest load $\hat{S}_N$ for various values of A, r, o, and N

2.3 Expected Number of Task Groups

Given the maximum number of task groups seen by an agent (N), we can calculate the expected heaviest agent load using Equation 1. But this begs the question of what is the maximum number of task groups an individual agent will see, given the actual (n) or expected number (η) that the entire system will see. The solution is similar—each agent either sees or does not see each of the n task groups, another binomial process. Let N_i be the number of task groups sensed by agent i, with a binomial distribution of parameters n and q. If a is again the number of agents that see a single task group and A the total number of agents, then $q = a/A$, the probability that each agent will see a particular track (we'll give an equation for a next). By the same derivation as in the last section, the max order statistic $\hat{N}$ has the probability function $g_{A,n,a/A}(s)$, and expected value:

$$E[\hat{N}|n, a] = \sum_{s=0}^{n} s g_{A,n,a/A}(s) \tag{2}$$

[4] R^2 is the squared correlation coefficient, a measure of goodness of fit. It may be interpreted as the proportion of total variabity in the observed data that is explained by the model.

Figure 3 shows the actual mean value of the maximum number of tracks seen by an agent over 1000 runs of the DSN simulation, versus the predicted value, for n tracks and square DSN organizations $[2 \leq r \leq 10, 1 \leq \sqrt{A} \leq 10, 1 \leq n \leq 10]$ without any overlap ($R^2 > .96$)

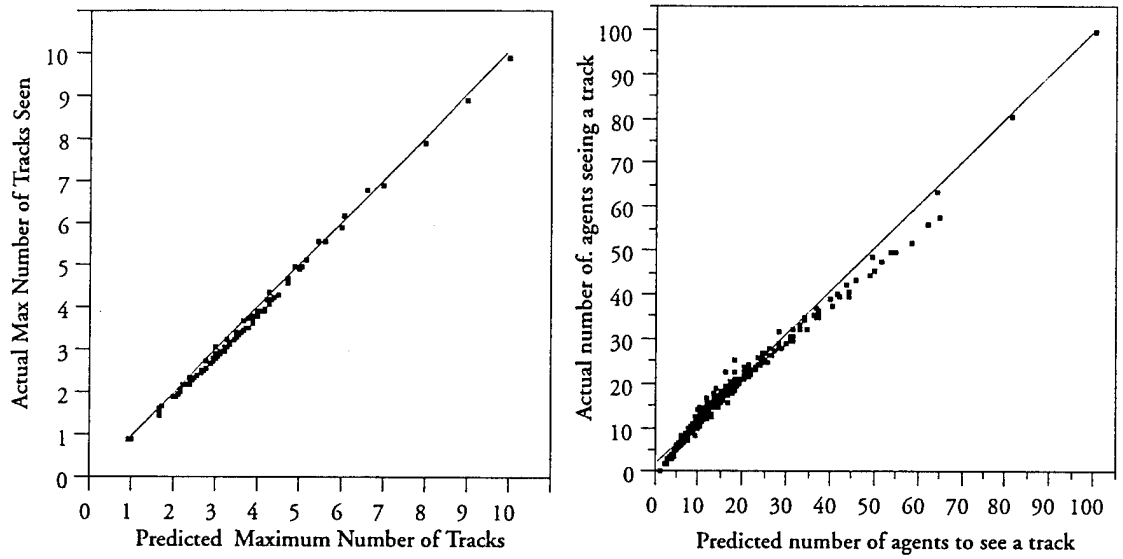


Figure 3: On the left, actual versus predicted maximum number of task groups (tracks) seen by any one agent for various r, A, and n. On the right, actual versus predicted average number of agents seeing a single task group (track) for various r, o, and A.

2.4 Expected Number of Agents

The only remaining term we need to analyze before deriving an expression for system performance is a, the expected number of agents that will see a single task group. In general, a will depend on the total number of agents A and the organization (r and o). When there is only one agent, it will see every task group ($a = 1$). When the agents overlap completely, every agent sees every task group ($[o = r] \rightarrow [a = A]$). When the agents in a square environment do not overlap, $a = \sqrt{A}$. The relationship follows the ratio of the area solely covered by an agent plus the area of the overlapping section, to the total area covered alone:

$$a = A^{(\frac{r^2+o^2}{2r^2})} \tag{3}$$

Note that a is not a random variable, it is just derived directly from environmental parameters. Figure 3 shows a regression of the actual average value of a over 1000 runs verses the predicted value for all 630 DSN organizations $[2 \leq r \leq 10, 0 \leq o \leq r, 1 \leq \sqrt{A} \leq 10]$ ($R^2 > .98$).

2.5 Work Involved in a Task Structure

Finally we turn to modeling the performance of the system as a whole, which is based on the structure of the tasks involved. We have developed a characterization of task environments that formally captures the range of features, processes, and especially interrelationships that occur during computationally intensive coordination and scheduling [5]. The model of environmental and task characteristics we propose has three levels: *objective*, *subjective*, and *generative*; the subjective level is not discussed here.

The *objective* level describes the essential structure of a particular problem-solving situation or instance over time. It focuses on how task interrelationships dynamically affect the *quality* and *duration* of each task. In this paper we will concentrate only on duration as a performance metric. The basic

model is that task groups $\mathcal{T}$ occur in the environment at some frequency, and induce tasks T to be executed by the agents under study. Task groups are independent of one another, but tasks within a single task group have interrelationships. An individual task that has no subtasks is called a method M and is the smallest schedulable chunk of work. The quality and duration of an agent's performance on an individual task is a function of the timing and choice of agent actions ('local effects'), and possibly previous task executions ('non-local effects'). When local or non-local effects exist between tasks that are known by more than one agent, they become *coordination relationships* [4, 3]. The basic purpose of the objective model is to formally specify how the execution and timing of tasks affect quality and duration.

At the lowest level, each method (leaf task) M at time t can produce, if executed, some maximum quality $\mathbf{q}(M,t)$ in some amount of time $\mathbf{d}(M,t)$ (each method has an initial maximum quality and duration $\mathbf{q_0}(M)$ and $\mathbf{d_0}(M)$). $Q(M,t)$ will denote the quality of method M at time t; $Q(M,t) = 0$ before a method is executed, and in the simplest case $Q(M,t) = \mathbf{q}(M,t) = \mathbf{q_0}(M)$ after the method is executed. In this paper, we will be concerned primarily with method durations, and the choice of method quality accrual function $Q(M,t)$ is not significant. Any task execution that starts before the execution of T completes may potentially affect T's execution through *non-local effects.* There are precisely two possible non-local effects on T under our model: **duration effects**, where $\mathbf{d}(T,t)$ (duration) is changed for some or all of the methods for a task; and **quality effects**, where $\mathbf{q}(T,t)$ (quality) is changed similarly. A non-local effect on a task is initiated by the execution of some other task in the task group. The effect is dependent on the relative timing of the two task executions, the quality of the task causing the effect, and whether information was transmitted between the two tasks.

Other work has considered the case of *facilitation*, a non-local effect where the availability of a result from one task alters the quality and duration of another task, but the non-communication of a result has no effect [4]. This work considers a different relationship, *precedence.* If task A precedes task B, then the maximum quality $\mathbf{q}(B,t) = 0$ until A is completed and the result is available, when the maximum quality will change to the initial maximum quality $\mathbf{q}(B,t) = \mathbf{q_0}(B)$.

2.5.1 Execution Model

For this paper we use an extremely simple model of execution. Agents can perform three actions: method execution, communication, and information gathering. The control component of an agent determines the next action an agent will perform based on the agent's current set of beliefs [1, 13]. A method execution action, of method M, that is begun at time t will conclude at time $t+\mathbf{d}(M,t)$. An information gathering action has duration $\mathbf{d_0}(I)$ and updates the agent's set of beliefs with any new information in the environment, for example, the arrival of data at the start of an episode, or communications from other agents. A communication action has duration $\mathbf{d_0}(C)$ and, after a communication delay, makes information (such method execution results) available to other agents. The agent on the receiving side must perform an information gathering action before the communication can affect its local beliefs.

2.5.2 Simple Objective DSN Model

Recall that the summary of a DSN environment was the tuple $\mathcal{D} =< A, \eta, r, o, \mathcal{T} >$; this will become our generative model, especially the parameter η (expected number of task groups). A particular episode in this environment can be described by the tuple $D =< A, r, o, \mathcal{T}_1, \ldots, \mathcal{T}_n >$ where n is a random variable drawn from an unknown distribution with location parameter (central tendency) of η. Note that we make almost no assumptions about this distribution; its characteristics will differ for different environments. For example, in the description of our DSN simulation early in Section 2 we noted the physical process by which vehicle tracks were generated and that the length of the tracks was not normally distributed.

Each task group $\mathcal{T}i$ is associated with a track of length l_i and has the same basic objective structure, based on the DVMT:

- l_i Vehicle Location Methods (VLM's) that represent processing raw signal data at a single location to a single vehicle location hypothesis.
- $l_i - 1$ Vehicle Tracking Methods (VTM's) that represent short tracks connecting the results of the VLM at time t with the results of the VLM at time $t + 1$.
- 1 Vehicle Track Completion Method (VCM) that represents merging all the VTM's together into a complete vehicle track hypothesis.

Non-local precedence effects exist between each method at one level and the appropriate method at the next level as shown in Figure 4—two VLMs precede each VTM, and all VTM'S precede the lone VCM.

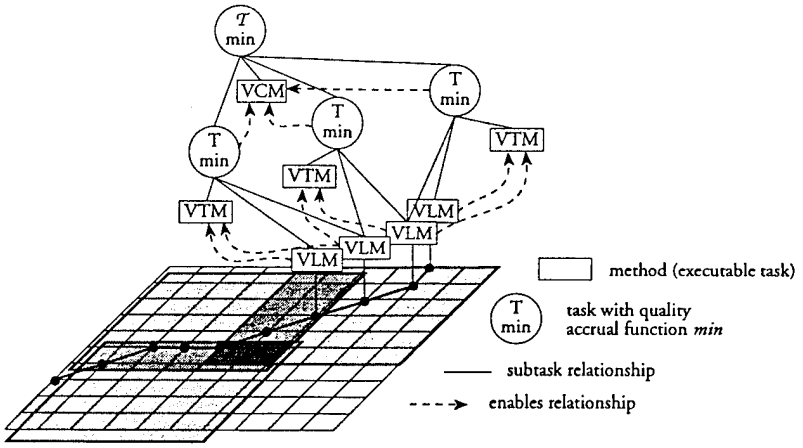


Figure 4: Objective task structure associated with a single vehicle track.

If we assume that each VLM has initial duration $\mathbf{d_0}(\text{VLM})$ and each VTM has the initial duration $\mathbf{d_0}(\text{VTM})$, then we can see from the task structure that for each task group the total execution time taken by a single processor agent will be:

$$l_i\mathbf{d_0}(\text{VLM}) + (l_i - 1)\mathbf{d_0}(\text{VTM}) + \mathbf{d_0}(\text{VCM}) \tag{4}$$

This task structure is a simplification of the real DVMT task structure. For example, there is no sensor noise (which will cause *facilitation* relationships between tasks, and there is no confusion caused by 'ghost tracks'. Adding these features to the task structure will cause some interesting phenomena that we will discuss briefly in the conclusions.

3 Model Summary

This section has described our basic model of the DSN environment, beginning with the basic parameters $\mathcal{D} =< A, \eta, r, o, \mathcal{T} >$. A particular episode in this environment can be indicated as $D =< A, r, o, \mathcal{T}_1, \ldots, \mathcal{T}_n >$, where each of the n task groups has the structure mentioned above. This section also developed expressions for $\hat{S}$, the number of low-level sensor subtasks (i.e., VLM's) at the most heavily loaded agent, the number of task groups $\hat{N}$ at the most heavily loaded agent, and the average number of agents a that see a single task group. Another way to look at these results is that we have derived the probability distributions of these variables, i.e., if the system of agents as a whole sees n total task groups, then the distributions of $\hat{N}$ and $\hat{S}$ are:

$$\Pr[\hat{N} = N|n] \;=\; g_{A,n,\frac{a}{A}}(N) \tag{5}$$

$$\Pr[\hat{\mathbf{S}} = s|\hat{N} = N] = g_{a,N,0.5}(s) \quad (6)$$

$$\hat{S} = (r\hat{\mathbf{S}} + (r/2)(N - \hat{\mathbf{S}})) \quad (7)$$

Finally Eq. 4 derived the duration, or amount of processing work, involved in a single task group from its structure.

In the next section we will combine these results with simple local scheduling and coordination algorithms appropriate for static and dynamic organizations. This will allow us to describe the performance of a system of agents following one of these organizational algorithms in a particular episode or environment.

4 Static vs. Dynamic Organizational Structures

Now we have the necessary background to analyze static and dynamic organizational structures. We have equations for the maximum expected number of subtasks at an agent given the number of task groups seen $\hat{S}$, the maximum expected number of task groups seen given the total number $\hat{N}$, and the predicted number of agents sensing part of a task group a. The key to static structures is to divide up the overlap area *a priori* (rather than to penalize agents for doing redundant work in the overlap area [7]). The key to dynamic organizational structures is to transfer tasks so that all the agents' resources are used efficiently.

We will repeat the assumptions we discussed at the start of Section 2 on page 1: the agents are homogeneous (have the same capabilities with respect to receiving data, communicating, and processing tasks), the agents are cooperative (interested in maximizing the system performance over maximizing their individual performance), the data for each episode is available simultaneously to all agents as specified by their initial organization, and there are only structural (precedence) constraints within the subtasks of each task group.

4.1 Analyzing Static Organizations

In a static organization, agents divide the overlapping areas of their ranges as evenly as possible. The result is a new area of responsibility $r' = r - \frac{o}{2}$ for each agent with no overlap (see Figure 5).[5] Given the task structure as described in Section 2.5 and shown in Figure 4, and any raw data or communicated task results provided by information gathering actions, the agent can at any time build a list of currently executable methods (under the set of precedence constraints). Also, at any time an agent can build a list of methods that need to be executed, but cannot be because their precedence constraints have not yet been met. The communication action in this algorithm is a broadcast of the highest level results of all the task groups an agent has worked on. Each agent follows the same control algorithm (remember, all the raw data is available at the start) and terminates when all task groups are completed (either locally or by reception of the result from another agent):

[5] The reason for overlap will be apparent in dynamic structures—multiple agents can work in an overlapping area without paying any cost for communicating raw data between them. Overlap can also provide redundancy in case of agent failure.

```
(Repeat
        Do Information-Gathering-Action
        (Repeat
                Let E = [get set of currently executable methods]
                (For method In E
                        Do Method-Execution-Action(method))
        Until (null E))
        Do Communication-Action(broadcast highest-level results)
        Let W = [get set of methods still waiting on precedence constraints]
Until (null W))
```

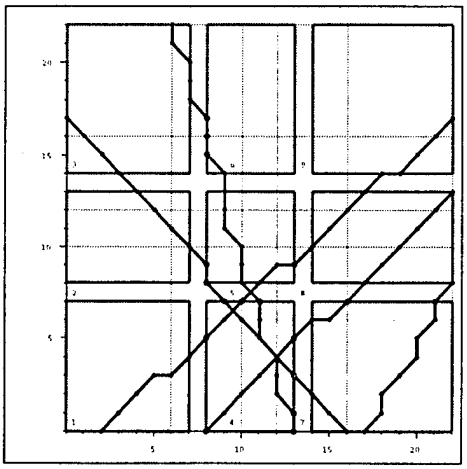


Figure 5: Example of a 3x3 organization, $r = 11$, $o = 5$, with 5 tracks. The thick dark grey boxes outline the default static organization, where there is no overlap.

First, let us analyze this algorithm assuming that only *one* task group (vehicle track) is present. In the environment $\mathcal{D} =< A, \eta, r, o, \mathcal{T} >$, if we let S' represent the largest amount of low-level data in *one* task group seen by any agent, and a the total number of agents that see the task group (from Equation 3), then the amount of time it will take that agent to construct a complete solution is equal to the amount of time it will take for the initial information gathering action ($\mathbf{d_0}(I)$) plus the amount of time to do all the local work ($S'\mathbf{d_0}(\text{VLM}) + (S' - 1)\mathbf{d_0}(\text{VTM})$), communicate that work ($\mathbf{d_0}(C)$), get the other agents' results ($\mathbf{d_0}(I)$), plus the amount of time to combine results from the other $a - 1$ agents ($(a-1)\mathbf{d_0}(\text{VTM})$), plus time to produce the final complete task group result ($\mathbf{d_0}(\text{VCM})$), plus communicate that result to everyone ($\mathbf{d_0}(C)$). For simplicity we will assume that $\mathbf{d_0}(I)$ and $\mathbf{d_0}(C)$ are constant and do not depend on the amount of data. Note also that since this agent is the most heavily loaded, by definition all other agents will finish their work and have communicated it by the time this agent finishes its local work.

In the general case, if the system sees $n = \eta$ total task groups, then the expected amount of low-level sensor data (size of the initial data set) at the maximally loaded agent can be derived from the marginal expected value for $\hat{S}$ given the joint distribution of $\hat{S}$ (Eqns. 6, 7) and $\hat{N}$ (Eq. 5):

$$E[\hat{S}] = \sum_{N=0}^{n} \sum_{s=0}^{N} g_{A,n,\frac{a}{A}}(N) g_{a,N,p}(s)(rs + \frac{r}{2}(N - s)) \tag{8}$$

Similar to the single task group case, the total time until termination for an agent receiving an initial data set of size $\hat{S}$ is the time to do local work, combine results from other agents, and build the completed results, plus two communication and information gathering actions:

$$\hat{S}\mathbf{d_0}(\text{VLM}) + (\hat{S} - \hat{N})\mathbf{d_0}(\text{VTM}) + (a-1)\hat{N}\mathbf{d_0}(\text{VTM}) + \hat{N}\mathbf{d_0}(\text{VCM}) + 2\mathbf{d_0}(I) + 2\mathbf{d_0}(C) \tag{9}$$

We can use Eq. 9 as a predictor by combining it with the probabilities for the values of $\hat{S}$ and $\hat{N}$ given in Eqns. 6, 7, and 5 (this is very similar to the treatment in Eq. 8).

We tested these predictions of Equation 9 versus the mean termination time of our DSN simulation over 10 repetitions in each of 43 randomly chosen environments from the design space $[2 \leq r \leq 10, 0 \leq o \leq r, 1 \leq \sqrt{A} \leq 5, 1 \leq N \leq 10]$. The durations of all tasks were set at 1 time unit, as were the duration of information gathering and communication actions; we will demonstrate and discuss the effect of this assumption later in the paper. We used the simulation validation statistic suggested by Kleijnen [9] (where $\hat{y}$ = the predicted output by the analytical model and y = the output of the simulation):

$$z = \frac{y - \hat{y}}{(\mathrm{Var}(y) + \mathrm{Var}(\hat{y}))^{1/2}} \tag{10}$$

where $\mathrm{Var}(\hat{y})$ is the predicted variance.[6] The result z can then be tested for significance against the standard normal tables. In each case, we were unable to reject the null hypothesis that the actual mean termination equals the predicted mean termination at the $\alpha = 0.05$ level. For non-statisticians: this is a good thing. The null hypothesis is that our prediction is the same as the actual value, we did not wish to reject it, and we did not. However, such a test has problems since there are other possible reasons which might prevent us from rejecting the null hypothesis. The best approach to avoiding this problem is to report not only the alpha level of the test but also the test's power. With any statistical hypothesis test there are four possible outcomes: we are unable to reject the null hypothesis when it is in reality true , we are unable to reject the null hypothesis when it is in reality false (called a *Type II error*), we reject the null hypothesis when it is in reality false, and we reject the null hypothesis when it is in reality true (called a *Type I error*). The α level is essentially the probability of a Type I error. The *power* of a test is $(1 - \beta)$, where β is essentially the probability of a Type II error (accepting a false null hypothesis). Unfortunately, estimating the power of a test is often difficult, and requires detailed knowledge of the distribution of the test statistic under the condition that the alternative hypothesis is true (this is turn usually requires the computation of the power for several likely alternative hypotheses). Future work will involve the computation of the power for this statistic, or the development of a new test based on other possible statistics that have better-understood power characteristics. Figure 6 shows the mean of 10 repetitions in each environment versus the expected value and its likelihood intervals. Thus the analytical model describes the implementation fairly well, and we could use the analytical model to design a good static organization for a given environment, using standard heuristic optimization techniques such as simulated annealing.

4.2 Control Costs

The control algorithm presented above is simple and not necessarily optimal. By communicating only when there is no local work to be done, the heaviest-loaded agent gives up the chance for other agents to do the high-level composition in a task group by incrementally transmitting each result (or set of results on a single task group) as it is completed—a maximum potential savings of $(\hat{N} - 1)(a - 1)\mathbf{d}_0(\mathrm{VTM}) + (\hat{N} - 1)\mathbf{d}_0(\mathrm{VCM})$. However, this needs to be balanced with the cost of multiple communication actions, which is $(\hat{N} - 1)\mathbf{d}_0(C)$. Thus the question of 'when to communicate' (when to incrementally transmit partial results) rests directly on the cost of communication relative to $(a - 1)\mathbf{d}_0(\mathrm{VTM}) + \mathbf{d}_0(\mathrm{VCM})$ (which depends on both the basic method durations and the agents' organizational structure).

This simple control algorithm can be analyzed easily, unlike many other systems where control costs are ignored. If we view the cost of control as the time spent by an agent when *not* performing an action

[6] The predicted variance of Equation 9 can be easily derived from the statistical identity $\mathrm{Var}(x) = E[x^2] - (E[x])^2$.

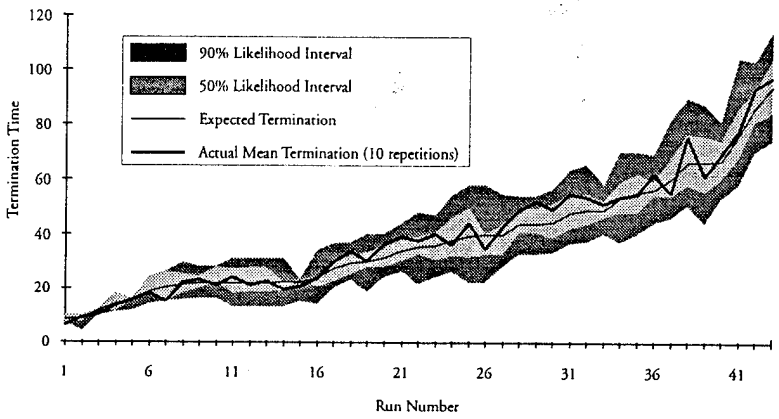


Figure 6: Actual system termination versus analytic expected value and analytically determined 50% and 90% likelihood intervals. Runs arbitrarily ordered by expected termination time.

(executing a method, information gathering, communication), then our algorithm runs in constant time between actions except for the two tests [*get set of currently executable methods*] and [*get set of methods still waiting*]. Each of these in the worst case requires a constant-time test of each element of the full task structure, which is of size $O(\eta r)$. Thus we see how the control costs, too, are related to organizational structure.

4.3 Analyzing Dynamic Organizations

In the dynamic organizational case, agents are not limited to the original organization and initial distribution of data. Agents can re-organize by changing the initial static boundaries (changing responsibilities in the overlapping areas), or by shipping raw data to other agents for processing (load balancing). We will assume in this section that the agents do not communicate about the current local state of problem solving directly (see Section 5 on using meta-level communication). A clearer distinction is that in the dynamic organization each agent makes its initial decision (about changing boundaries or shipping raw data) without access to non-local information. In the meta-level communication situation discussed later the agent has access to both its local information and a summary of the local state of other agents. In either case the decision to dynamically change the organization is made only once, at the start of an episode.

In the case of reorganized overlapping areas, agents may shift the initial static boundaries by sending a (very short) message to overlapping agents, telling the other agents to do all the work in the overlapping areas. The effect at the local agent is to change its effective range parameter from its static value of $r' = r - o/2$ to some value r'' where $r - o/2 \geq r'' \geq r - o$, changing the first two terms of Equation 9, and adding a communication action to indicate the shift and an extra information gathering action to receive the results. Section 4.4 discusses a particular implementation of this idea that chooses the partition of the overlapping area that best reduces expected differences between agent's loads and averages competing desired partitions from multiple agents.

In the second case, an agent communicates some proportion ρ of its initial data to a second agent, who does the associated work and communicates the results back. Instead of altering the effective range and overlap, this method directly reduces the first two terms of Equation 9 by the proportion ρ. The proportion ρ can be chosen dynamically in a way similar to that of choosing where to partition the overlap between agents (Section 4.4).

Whether or not a dynamic reorganization is useful is a function of both the agent's local utility and also the load at the other agent. We will again be concentrating on the agent with the heaviest load.

Looking first at the local utility, to do local work under the initial static organization with n task groups, the heaviest loaded agent will take time:

$$S'\mathbf{d}_0(\text{VLM}) + (S' - n)\mathbf{d}_0(\text{VTM}) \tag{11}$$

When the static boundary is shifted before any processing is done, the agent will take time

$$\mathbf{d}_0(C_{\text{short}}) + S''\mathbf{d}_0(\text{VLM}) + (S'' - n)\mathbf{d}_0(\text{VTM}) + \mathbf{d}_0(I) \tag{12}$$

to do the same work, where C_{short} is a very short communication action which is potentially much cheaper than the result communications mentioned previously, and S'' is calculated using r''. When balancing the load directly, local actions will take time

$$\mathbf{d}_0(C_{\text{long}}) + \rho S'\mathbf{d}_0(\text{VLM}) + \rho(S' - n)\mathbf{d}_0(\text{VTM}) + \mathbf{d}_0(I) \tag{13}$$

where $\mathbf{d}_0(C_{\text{long}})$ is potentially much more expensive than the communication actions mentioned earlier (since it involves sending a large amount of raw data). If the other agent had no work to do, a simple comparison between these three equations would be a sufficient design rule for deciding between static and either dynamic organization.

Of course, we cannot assume that the other agent is not busy; the best we can do *a priori* (without meta-level communication during a particular episode) is to assume the other agent has the *average* amount of work to do. We can derive *a priori* estimates for the average initial work at another agent from Equation 9 by replacing the probability function of the max order statistic $g_{a,N,p}(s)$ with the simple binomial probability function $b_{N,p}(s)$. Therefore without any meta-level communication, a system of agents could choose intelligently between static, dynamic overlap reorganization, and dynamic load balancing given these constraints.

4.4 Dynamic Coordination Algorithm for Reorganization

This section describes a particular implementation of the general idea described earlier of dynamically reorganizing the partitions between agents for the DSN simulation. This implementation will keep each agent's area of responsibility rectangular, and relaxes competing constraints from other agents quickly and associatively (the order of message arrival does not affect the eventual outcome). To do this, the message sent by an agent requests the movement of the four *corridors* surrounding an agent. The northern corridor of Agent 1, for example, is the northern agent organizational responsibility boundary shared by every agent in the same *row* as Agent 1. As can be seen in Figure 7, a 3x3 organization has four corridors (between rows 1 and 2, 2 and 3, and between columns 1 and 2, 2 and 3).

The coordination algorithm described here works with the local scheduling algorithm described earlier in Section 4.1. This is consistent with our view of coordination as a *modulating* behavior [4]. The only modification to the local scheduler is that we prevent it from scheduling local method execution actions until our initial communications are completed (the *initial* and *reception* phases, described below).

The coordination algorithm is then as follows. During the *initial* phase the local scheduler schedules the initial information gathering action, and we proceed to the second phase, *reception.* In the second phase we use the local information to decide what organizational design to use, and the parameter values for the design we choose. To do this we calculate the duration of our (known) local work (Eq. 11), and then estimate that duration under the alternative organizations (dynamic reorganization or load-balancing). When a parameter needs to be estimated, we do so to minimize the absolute expected difference between the amount of work to be done locally and the amount of work done at the remote agent that is impacted the most by the proposed change.

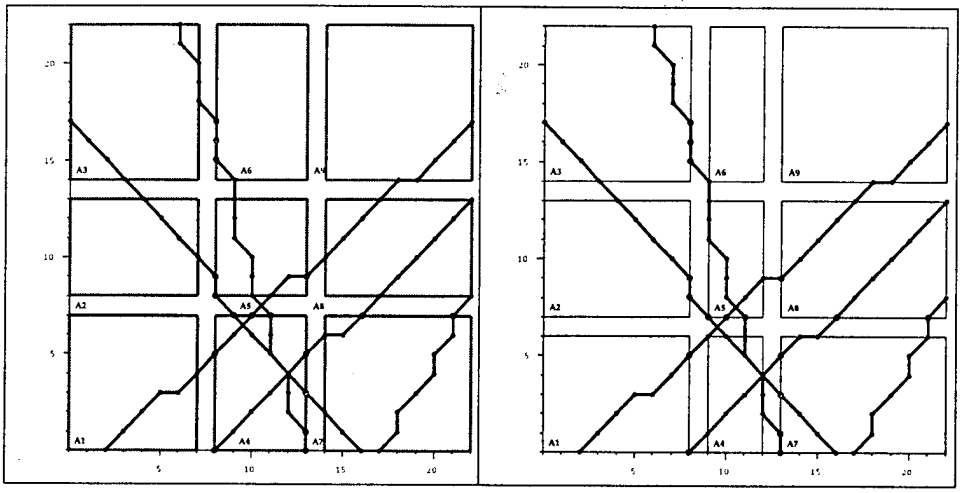


Figure 7: On the left is a 3x3 static organization, on the right is the dynamic reorganization result after agents 3, 4, 5 and 7 attempt to reduce their areas of responsibility by one unit. In this example the corridors running North to South have been moved closer by two units to reduce the load on agents 4, 5, and 6 in the second column.

For example, when dynamically restructuring, if the overlap between agents is more than 2 units, we have a choice of reducing the area an agent is responsible for by more than 1 unit (this is the organizational design parameter ρ in question). To decide on the proper reduction (if any), each agent computes its known local work $\hat{W}(\rho)$ using Eq. 11 with the actual (not estimated) S' and N computed assuming the agent's area is reduced by ρ. Then the agent finds the value of ρ that minimizes the difference in its known local work $\hat{W}(r-\rho)$ and the *average* work $\bar{W}(r+\rho)$ at the other agent:

$$
\begin{aligned}
S(r,s,N) &= (rs + \frac{r}{2}(N-s)) \\
W(r,s,N) &= S(r,s,N)\mathbf{d_0}(\mathrm{VLM}) + (S(r,s,N)-N)\mathbf{d_0}(\mathrm{VTM}) \\
E[\hat{W}(r)] &= \sum_{N=0}^{n}\sum_{s=0}^{N} g_{A,n,\frac{a}{A}}(N)\, g_{a,N,p}(s)\, W(r,s,N) \qquad (14) \\
E[\bar{W}(r)] &= \sum_{N=0}^{n}\sum_{s=0}^{N} b_{n,\frac{a}{A}}(N)\, b_{N,p}(s)\, W(r,s,N) \qquad (15)
\end{aligned}
$$

Equation 15 is just a restatement of Eqn. 14 (itself derived from Eqns. 5, 6, 7, and 11 for the case of the average, not maximally loaded, agent (thus the use of b, the binomial probability function rather than g, the max order statistic p.f.).

If $\rho = 0$, then the agent will not restructure. If $\rho \neq 0$, then the agent sends a message to all affected agents requesting a reduction of amount ρ in each corridor (north, east, south, and west). The agent sets its current area of interest to include only the unique (non-overlapping) portion of its area (if any), and enters the *unique-processing* phase. During this phase the regular local scheduler described earlier controls method execution actions.

When no more methods unique to this agent can be executed, the coordination algorithm checks the current time. If enough time has passed for the messages from other agents (if any) to arrive (this depends on the communication delays in the system), the coordination algorithm schedules an information-gathering action to retrieve the messages. Note that every agent may reach this point at a different time; agents with a large amount of unique local work may take some time, agents with no work at all will wait idle for the length of communication delay time in the system.

At this point each agent will relax its borders according to the wishes of the other agents. The relaxation algorithm we have chosen is fairly simple and straightforward, though several similar choices are possible. The algorithm is symmetric with respect to the four corridors surrounding the agent, so we will just discuss the relaxation of the northern corridor. There will be a set of messages about that corridor, some wanting it moved up by some amount and some wanting it moved down by some amount—we will consider these as positive and negative votes of some magnitude. The relaxation algorithm sums the votes, and returns the sum unless it is larger than the maximum vote or smaller than the minimum vote, in which case the max or min is returned, respectively. At this point the agent enters the final *normal processing* phase, and the local scheduler schedules all further actions as described earlier.

4.5 Analyzing the Dynamic Restructuring Algorithm

As we did in Section 4.1, we can develop an expression for the termination time of any episode where the agents follow this algorithm. To do so, we start with the basic termination time given all of the random variables. This equation is derived from Eqns. 14 and 15:

$$T(r, \hat{S}, \hat{N}, \bar{s}, \bar{N}) = \max[W(r - \rho, \hat{S}, \hat{N}), W(r + \rho, \bar{s}, \bar{N})] \tag{16}$$

where ρ is computed as described in the last section using the given values of $(r, \hat{S}, \hat{N}, \bar{s}, \bar{N})$. To turn this into a predictive formula, we then use the expressions for the probabilities of the terms $\hat{S}$, $\hat{N}$, $\bar{s}$, and $\bar{N}$ (from Eqns. 6, 7, and 5). For example, we can produce an expression for the expected termination of the algorithm:

$$\sum_{\hat{N}=0}^{n} \sum_{\hat{S}=0}^{\hat{N}} \sum_{\bar{N}=0}^{n} \sum_{\bar{s}=0}^{\bar{N}} g_{A,n,\frac{o}{A}}(\hat{N}) g_{a,\hat{N},0.5}(s) b_{n,\frac{o}{A}}(\bar{N}) b_{\bar{N},0.5}(\bar{s}) T(r, \hat{S}, \hat{N}, \bar{s}, \bar{N}) \tag{17}$$

We tested the predictions of Equation 17 versus the mean termination time of our DSN simulation over 10 repetitions in each of 10 randomly chosen environments. The durations of all tasks were set at 1 time unit, as were the duration of information gathering and communication actions, with the exception of the 4 environments described in the next section. Using the same validation statistic as before (Eq. 10) in each case we were unable to reject the null hypothesis that the actual mean termination equals the predicted mean termination at the $\alpha = 0.05$ level.[7]

4.5.1 Increasing Task Durations

Figure 8 compares the termination of static and dynamic restructuring organizations on identical episodes in four different environments. From left to right, the environments were $[A = 9, r = 9, o = 9, n = 7]$, $[A = 4, r = 9, o = 3, n = 5]$, $[A = 16, r = 8, o = 5, n = 4]$, $[A = 9, r = 10, o = 6, n = 7]$. Ten different episodes were generated for each environment. In order to see the benefits of dynamic restructuring more clearly, we chose task durations for each environment similar to those in the DVMT: $\mathbf{d_0}(\text{VLM}) = 6, \mathbf{d_0}(\text{VTM}) = 2\mathbf{d_0}(\text{VCM}) = 2$.[8] Note that the dynamic organization often does significantly better than the static organization, and rarely does much worse—remember that is many particular episodes that the dynamically organized agents will decide to keep the static organization, although they pay a constant overhead when they keep the static organization (one extra communication action and one extra information gathering action, given that the time for a message to reach all agents is no longer than the communication action time).

[7] The same caveats we discussed earlier still apply.

[8] The idea being that the VLM methods correspond to lowest three DVMT KSIs as a group, and the other methods correspond to single DVMT KSIs, and that a KSI has twice the duration of a communication action.

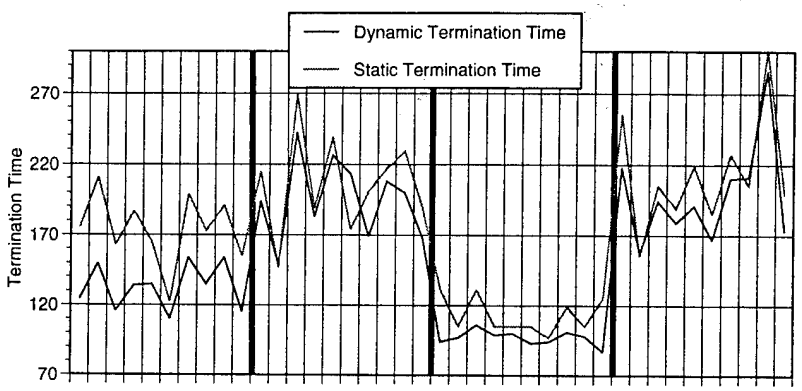


Figure 8: Paired-response comparison of the termination of static and dynamic systems in four different environments (ten episodes in each). Task durations are set to simulate the DVMT (see text).

5 Using Meta-Level Communication

For some environments $\mathcal{D} =< A, \eta, r, o, \mathcal{T} >$ one of the three organizational choices may be clearly better in the long run, but for most environments the choice is not so clear given the variance in system performance. The choice that optimizes performance over the long run is often not optimal in any particular episode. Taking the essential equations for local work in Section 4.3, we can compute likelihood intervals on the predicted performance of an organization under each of the three coordination regimes by combining the local likelihood interval on the expected load of the heaviest loaded agent, and the likelihood interval on the average agent load. These results, for the 50% likelihood interval, are shown in Figures 9 and 10. Again we have assumed that all execution, communication, and information gathering action durations have the same value (making communication relatively expensive). The first figure, Figure 9, highlights how the relationship between performance under a static organization and a dynamically load balanced organization changes as the number of agents increases. As expected, load balancing becomes more desirable as the number of agents increases (in relation to the average number of tracks): when there are many agents, the average agent load becomes very low, which offsets the cost of transferring tasks. In this figure the performance difference between static and overlap reorganization remains nearly constant relative to the number of agents.

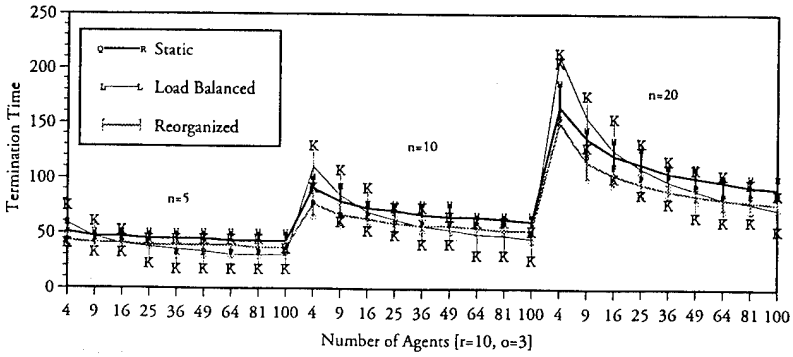


Figure 9: 50% likelihood intervals on the expected termination of a system under three coordination regimes, different numbers of agents, and three values of n

The second, Figure 10, points out how dynamically reorganizing the overlap area increases the performance over static organization as the amount of overlap increases. For this graph we assumed that the agents would shrink their entire area of responsibility (as opposed to minimizing the difference in

maximum versus average work as described in Section 4.4). This graph shows the need for dynamically calculating the shrinkage parameter (ρ) especially at high levels of overlap (note how the dynamically reorganized organization is predicted to do worse at high levels of overlap in the $n = 20$ portion). The expected performance difference between the static organization and load balancing remains relatively constant across changing values of o. In both figures we have let $\mathbf{d_0}(C) = \mathbf{d_0}(C_{\mathbf{short}}) = \mathbf{d_0}(C_{\mathbf{long}})$; increasing the differences in these values will move the corresponding curves directly up or down.

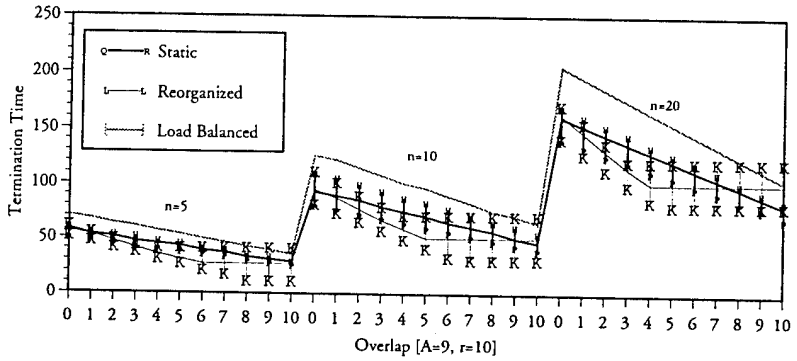


Figure 10: 50% likelihood intervals on the expected termination of a system under three control regimes, different overlaps, and three values of n. Likelihood intervals on the load balanced line are omitted for clarity.

The next series of figures demonstrate the effect of the ratio of computation duration to communication duration. This and subsequent figures assume that the dynamic restructuring shrinkage parameter ρ is set to minimize the difference between maximum and average local work as described in Section 4.4. Figure 11 shows how the expected value and 90% likelihood interval on system termination changes as the duration of a method execution action changes from equal to (1x) a communication action to 10 times (10x) that of a communication action. The task structure remains than of the DSN example described in Section 2.5. In Figure 11 we see a clear separation emerge between static and dynamic termination.

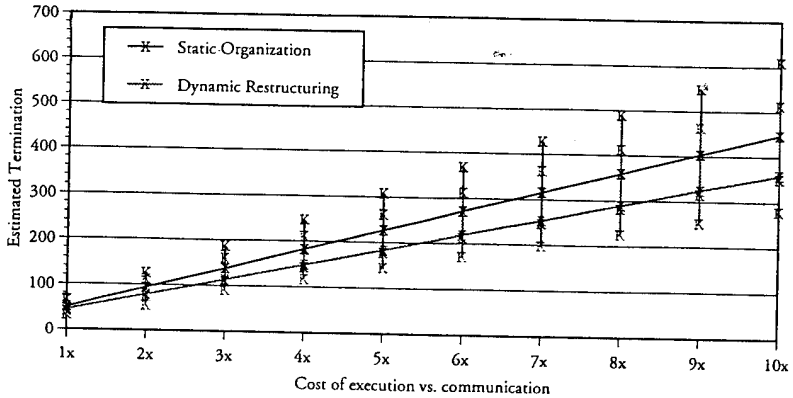


Figure 11: Effect of decreasing communication costs on expected termination under a static organization and dynamic restructuring (expected value and 90% likelihood interval, $A = 25, r = 9, o = 9, n = 7$).

These figures assume that the number of task groups n is known beforehand. The reason for this is to highlight the variance implicit in the organization, and minimize the influence of the external environment. Figure 12 shows how much extra variance is added when only the expected value of n, which is η, is known. We assume that the number of task groups n (in the DSN example, vehicle tracks) that occur during a particular episode has a Poisson distribution with an expected value of η.

The discrete probability function for the Poisson distribution, given in any statistics book, is then:

$$p_\eta(y) \quad = \quad \frac{\eta^y}{y!} e^{-y} \quad [\Pr[n = y]]$$

We can use this probability in conjunction with Eqns. 8, 11, and 15 to calculate the expected value, 50%, and 95% likelihood intervals on termination in the static or dynamic organizations. Note in Figure 12 both the large increase in variance when n is random (between the top bar and the third bar in the figure), and more importantly the small decrease in variance in the dynamic restructuring organization (between the first and the second bars). Note also that the mean termination time for the dynamic organization is slightly less than that for the static organization.

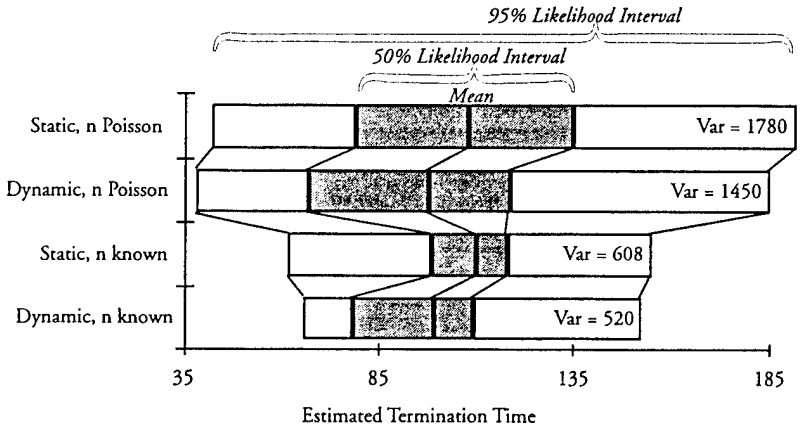


Figure 12: Demonstration of both the large increase in performance variance when the number of task groups n is a random variable, and the small decrease in variance with dynamic restructuring coordination $[A = 9, r = 9, o = 9]$. Where n is known, $n = 7$. Where n is a random variable, the expected value $\eta = 7$.

These figures bring us to the final point of this paper: often system performance can be improved significantly by dynamic reorganization, but it will rarely *always* be improved. Therefore, meta-level communication between agents about their local loads can, with a small communication cost, pinpoint the true costs and benefits of the various organizational structures, allowing an informed organizational decision to be made. Instead of an agent making a decision about restructuring or load balancing by assuming the *average* load, the agent will have the *actual* load for the neighboring agents. As we said in the introduction, the proper organization is often one that exploits *information that resolves uncertainties* about the current environment as it becomes available to the agents, allowing the agents to then create the most efficient organization for the situation.

6 Conclusions

The results of this paper can be looked at from three points of view. From the practitioner's viewpoint, the analysis presented here resulted in a set of design equations that can be used directly to optimize the performance of a simple DSN, or explore the design space given some model of how expensive agents are and what bounds (mean, median, 90% quantile) on their performance are required. Several of the simplifying assumptions we used, such as constant communication and information gathering costs, can be easily replaced with submodels chosen by the designer. From the viewpoint of the distributed AI community, we have returned to look at the some of the problems first studied by Durfee, Lesser, and Corkill[7]. They concluded that "Our intent is to show that overly specialized organizational structures allow effective network performance in particular problem-solving situations, but that no

such organization is appropriate in all situations." In this paper we reach the same abstract conclusion, but also show precisely what the effect is of a particular organizational structure (characterized by both its structural components and its coordination algorithm) in an environment (characterized by the structure and frequency of its tasks) in a clear way that not only allows us to predict performance but to explain it. The technique of using binomial approximations should also prove useful in different domains. From the viewpoint of the general research community this paper presents a methodology for answering questions about the design of a system by analysis and simulation. In such a methodology, the observation of particular phenomena in a complex system (the DVMT) leads to the building and verification of general models that predict and explain such phenomena.

In the short term, this work leads to the explanation of other interesting distributed problem solving phenomena displayed in [7]. The addition of noise at DSN sensors leads to the necessity of more complex coordination with the introduction of more complex task interrelationships (such as *facilitation*[4]). The addition of correlated noise in the environment can then cause these new, more complex coordination mechanisms to break down, producing the phenomenon recognized as *distraction.* In the long term, we are working towards a complete characterization of generalized partial global planning[3] as a first step towards a theory of coordination in distributed problem solving. We and other researchers are also using our task structure characterization for the analysis and simulation of problems in real-time and parallel scheduling.

References

[1] Philip R. Cohen and Hector J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42(3):213–261, 1990.

[2] R. Davis and R. G. Smith. Negotiation as a metaphor for distributed problem solving. *Artificial Intelligence*, 20(1):63–109, January 1983.

[3] Keith S. Decker and Victor R. Lesser. Generalizing the partial global planning algorithm. *International Journal of Intelligent and Cooperative Information Systems*, 1(2):319–346, June 1992.

[4] Keith S. Decker and Victor R. Lesser. Analyzing a quantitative coordination relationship. *Group Decision and Negotiation*, 2(3):195–217, 1993.

[5] Keith S. Decker and Victor R. Lesser. Quantitative modeling of complex computational task environments. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 217–224, Washington, July 1993.

[6] E. H. Durfee and T. A. Montgomery. Coordination as distributed search in a hierarchical behavior space. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(6):1363–1378, November 1991.

[7] Edmund H. Durfee, Victor R. Lesser, and Daniel D. Corkill. Coherent cooperation among communicating problem solvers. *IEEE Transactions on Computers*, 36(11):1275–1291, November 1987.

[8] J. Galbraith. *Organizational Design.* Addison-Wesley, Reading, MA, 1977.

[9] Jack P. C. Kleijnen. *Statistical Tools for Simulation Practitioners.* Marcel Dekker, New York, 1987.

[10] Victor R. Lesser and Daniel D. Corkill. The distributed vehicle monitoring testbed. *AI Magazine*, 4(3):63–109, Fall 1983.

[11] Terry M. Moe. The new economics of organization. *American Journal of Political Science*, 28(4):739–777, November 1984.

[12] Jasmina Pavlin. Predicting the performance of distributed knowledge-based systems: A modeling approach. In *Proceedings of the Third National Conference on Artificial Intelligence*, pages 314–319, Washington, D.C., August 1983.

[13] Yoav Shoham. AGENT0: A simple agent language and its interpreter. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, pages 704–709, Anaheim, July 1991.

[14] Arthur L. Stinchcombe. *Information and Organizations.* University of California Press, Berkeley, CA, 1990.

A Distribution of the Binomial Max Order Statistic

To expand on the formulae in Section 2.2, the amount of a single task group seen by an agent can be viewed as a random variable S with probability density function $f(s)$ and corresponding cumulative distribution function $F(s)$. Let a be the number of agents that initially see part of a single task group. By elementary statistics, the density of the max order statistic $S_{max} = \max(S_1, S_2, \ldots, S_a)$ is $g_a(s) = aF(s)^{a-1}f(s)$. The expected heaviest load then is $\int_0^r sg_a(s)ds$. In the DSN environment, S is discrete, and its probability function (determined empirically) is heavily weighted toward r (the maximum). To simplify the analysis, instead of letting S correspond to the number of subtasks in a single task group seen by an agent, we have a new random variable $\mathbf{S}$ equal 1 if the agent sees the maximum amount, and 0 otherwise. Now $\mathbf{S}$ has a Bernoulli (coin-tossing) distribution with parameter p corresponding to the chance of an agent seeing the maximum amount r of a task group. $\mathbf{S}$ then corresponds to the number of times an agent sees the maximum r if the agent sees N task groups (tracks). $\mathbf{S}$ has a binomial distribution with parameters p and N. Now if a agents see N task groups each, what is the distribution of $\hat{\mathbf{S}} = \max(\mathbf{S}_1, \mathbf{S}_2, \ldots, \mathbf{S}_a)$? This is the max order statistic for the binomial distribution, and because it is discrete, can be easily derived. The probability function and cumulative distribution function for the binomial distribution are:

$$\begin{array}{rcll} b_{N,p}(s) & = & \binom{N}{s}p^s(1-p)^{N-s} & [\Pr[\mathbf{S}=s]] \\ B_{N,p}(s) & = & \sum_{x=0}^{s} b_{N,p}(x) & [\Pr[\mathbf{S}\leq s]] \end{array}$$

If we assume each track is independent of the others, we can derive the cumulative distribution function:

$$\begin{aligned} \Pr[\hat{\mathbf{S}} \leq s] &= \Pr[\max(\mathbf{S}_1, \mathbf{S}_2, \ldots, \mathbf{S}_a) \leq s] \\ &= \Pr[\mathbf{S}_1 \leq s]\Pr[\mathbf{S}_2 \leq s]\cdots\Pr[\mathbf{S}_a \leq s] \\ &= B_{N,p}(s)^a \end{aligned}$$

The probability function $g_{a,N,p}(s) = \Pr[\hat{\mathbf{S}} = s]$ is then:

$$g_{a,N,p}(s) = B_{N,p}(s)^a - B_{N,p}(s-1)^a$$

APPENDIX D

A Formal Analysis of Solution Quality in FA/C Distributed Sensor Interpretation Systems*

Norman Carver
Computer Science Department
Southern Illinois University
Carbondale, IL 62901
(carver@cs.siu.edu)

Victor Lesser
Computer Science Department
University of Massachusetts
Amherst, MA 01003
(lesser@cs.umass.edu)

Abstract

The functionally-accurate, cooperative (FA/C) distributed problem-solving paradigm is one approach for organizing distributed problem solving among homogeneous, cooperating agents. The idea behind the FA/C model is that agents should produce tentative, partial results based on only local information and then exchange these results, exploiting the constraints that exist among their local subproblems to resolve the uncertainties and global inconsistencies that result from the use of incomplete information. While several FA/C systems have been implemented, there has been little formal analysis of the quality of the solutions that can be produced using the approach or of the conditions that are necessary for the approach to be effective. This paper reports on work we have done to formally analyze the FA/C model in the context of distributed sensor interpretation (SI). Several results are presented that compare the quality of solutions produced by a distributed FA/C system to those produced by an equivalent centralized system, based on particular agent problem-solving and coordination strategies. We first establish that while it is possible for an FA/C system to produce the same solution as a centralized system, this requires the use of interpretation and coordination strategies that are impractical for most SI applications. Because of this we then consider the effect of "approximate" interpretation and coordination strategies, given some assumptions about the characteristics of the domain.

* This work was supported in part by the Department of the Navy, Office of the Chief of Naval Research, under contract N00014-92-J-1450. The content of the information does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred.

1 Introduction

In the *functionally accurate, cooperative* (FA/C) paradigm for distributed problem solving [8, 10], agents need not have all the information necessary to completely and accurately solve their subproblems. Instead, agents are designed to produce tentative, partial results based on only local information and to then exchange these results with the other agents to resolve local uncertainties and global inconsistencies. The basic intuition behind this approach is that for many applications there exist (inter-agent) constraints among the subproblems, and these constraints can be exploited to resolve the inconsistencies and uncertainties that occur in local problem solving due to the lack of accurate, complete, and up-to-date information.

While several systems that use the FA/C approach have been built (e.g., [3, 9]), there has never been any formal analysis of the quality of the solutions that can be produced by the approach or of the effect of domain conditions and coordination strategies on solutions. This paper reports on our efforts in formally characterizing the FA/C paradigm. We will present some results about the quality of solutions that can be produced by an FA/C-based system relative to particular agent problem solving and coordination strategies. Since most FA/C applications have been in distributed sensor interpretation (SI) (particularly distributed vehicle monitoring) our analysis focuses on this domain.[1] Though we concentrate on this domain, we believe that our basic results will be applicable to other domains in which local problem solving involves approximate, satisficing search.

First, it is important to be clear about what we will and will not consider. [10] referred to the issues of *solution uncertainty* and *control uncertainty* in FA/C problem solving, and these concepts are useful in delineating the focus of our efforts. In this paper we concentrate on the issue of solution uncertainty: uncertainty about the correctness of partial solutions due to agents having incomplete local information. Control uncertainty is largely concerned with what actions agents should take and when they should take them to achieve efficient, coherent global problem solving. *Coordination* of agent activities involves both solution and control uncertainty, but we will ignore the issue of when information should be communicated and the possible communication of meta-level information. Thus, while we examine the conditions that are *necessary* for effective FA/C problem solving, these conditions are not sufficient to guarantee it. Coordination strategies for *efficient* FA/C problem solving remain an important area of research.[2] Another important issue in the design of FA/C systems is the role that agent architectures play in supporting effective FA/C problem solving. For example, some architectures allow a wide range of inconsistencies to be resolved using very directed, limited communication of data and hypotheses among the agents [3, 4]. Finally, there are other

[1] Distributed sensor interpretation plays an important role in many military situation assessment (decision support) systems.

[2] We are pursuing both empirical and analytic approaches to address this issue [6, 7].

factors that might prevent effective FA/C problem solving in certain domains, but they are beyond the scope of this paper.[3]

The main aspect of FA/C performance that we address is the quality of solutions produced by an FA/C system. Several theorems are presented that compare the solutions of an FA/C system to the solutions that would be produced by an "equivalent" centralized system, given particular agent problem-solving and coordination strategies. A key issue in this evaluation is the fact that approximate, satsificing techniques are often required to identify SI solutions (see Section 3). Because of this, there are several standards against which we might judge solution quality. We have chosen to define the "globally best solution" as the solution that would be produced by an *equivalent* centralized system: a centralized system using the same interpretation strategy as the individual distributed agents use for processing their own local data. The advantage of this definition is that it partially separates the effects of approximate local strategies from the effect of distributed processing.

We also consider the effect of different coordination strategies, in terms of the completeness with which subproblem (local interpretation) interactions are pursued. The analysis assumes the capabilities of an abstract model of the DRESUN system for distributed SI [3, 4]. In particular, we assume that all possible subproblem interactions can be easily identified and that they can be selectively (and incrementally) pursued (see Section 4). Again, there are several alternative strategies that might be used. These range from "complete strategies" that effectively eliminate the effects of the distribution of data, to "incomplete strategies" that do not.

Our results begin by showing that it is possible for a DRESUN-based distributed SI system to produce exactly the same global interpretation as would be produced by a centralized system and that this can indeed be the optimal solution. However, this is true only when the centralized system and the agents are doing complete, exact interpretation and subproblem interactions are completely resolved. As we point out, these are unrealistic assumptions for SI in general and for distributed SI. A key assumption of previous FA/C research has been that solutions can be produced without the need for "excessive" communication among the agents. This is the main reason for using incomplete coordination strategies: to make FA/C problem solving efficient enough to be effective. The rest of our results focus on the effects of incomplete, approximate coordination strategies.

In exploring the use of incomplete coordination strategies, the most important issue we will explore is whether FA/C systems can focus on the subproblem interactions involving only the local solutions produced by the agents. This can greatly increase the efficiency of FA/C problem solving. Unfortunately, this strategy does not in general guarantee that the global solution would be be

[3] For example, in some domains it could be substantially more difficult to represent and develop *partial* solutions than it would be to deal with only complete solutions.

the optimal solution or even the same as what would be produced by a centralized system. In conjunction with this result, we discuss certain domain characteristics that would allow an FA/C distributed SI system to produce solutions that are "satisfactory." The key concept that we consider is what we term *approximate monotonicity.*

The final set of results focus on the use of approximate interpretaton and coordination strategies where solutions are selected using a satisficing approach in which hypotheses are judged part of the solution if their belief surpasses an "acceptance threshold." This is a common strategy for many SI systems. Here the coordination strategy is incremental in that it uses the same criteria to determine the degree to do pursue additional inter-agent suproblem interactions or not. We show that there are conditions under which it is possible to guarantee that the distributed system produces a solution that is comparable to the centralized solution, and other conditions under which there is merely some probability of obtaining such a solution.

In the next two sections we describe our model of (distributed) sensor interpretation, discuss why SI is different from much studied "diagnosis problems," and then examine the use of approximate strategies for determining SI solutions. Section 4 introduces the DRESUN distributed SI model that we assume for our analysis, followed in Section 5 by a description of the coordination strategies that will be considered. Section 6 then defines the notation that will be used in the remainder of the paper. Section 7 contains the theorems and proofs related to FA/C solution quality for distributed SI. The paper concludes with a summary of the major results in the paper and our future research plans.

2 Distributed Sensor Interpretation

By sensor interpretation, we mean the determination of high-level, conceptual explanations of sensor data. Our model of the interpretation process will be essentially that used in [2]: interpretation hypotheses are incrementally constructed via *abductive inferences*, based on a causal domain model that defines the relationships among the data types and abstraction types. For each type T, the causal model defines the type's support, $\mathcal{S}_T$, and its possible explanations, $\mathcal{E}_T$. $\mathcal{S}_T = \{\mathsf{S}_i : \mathsf{S}_i$ is a *type instance specification*, which defines a legal support for a hypothesis of type $T\}$ (a type instance specification has an associated interpretation type and a set of constraints that define the legal attribute values for hypotheses of that type to support a hypothesis of type T). $\mathcal{E}_T = \{\mathsf{E}_i : \mathsf{E}_i$ is an interpretation type that might explain some type T hypothesis$\}$. If T is a *data* type then $\mathcal{S}_T$ is empty. If T is a *top-level* type then $\mathcal{E}_T$ is empty.

An interpretation system makes abductive inferences that identify possible explanations for a piece of data or a hypothesis. Thus, every hypothesis h with $type(h) = T$ is the result of a set of abductive inferences each of the form $h_i \Rightarrow h$, where $type(h_i) \in \mathcal{S}_T$. Conversely, each hypothesis

h with $type(h) = T$ may be part of a set of abductive inferences of the form $h \Rightarrow h_j$, where $type(h_j) \in \mathcal{E}_T$. We will use the following notation: *support-of*$(h) = \{h_i\}$, the (immediate) support for h; $supports(h, h_i)$; and $explains(h, h_j)$. In addition to its immediate support, each hypothesis must ultimately be supported by sensor data via a chain of abductive inferences $d \Rightarrow h_1 \Rightarrow \ldots \Rightarrow h_m$, where d is data and $supports(h_1, d)$, $supports(h_l, h_{l-1})$. To deal with these inference chains, we will use the notation *support-of**(h) to refer to that data that ultimately supports h, as well as the analogous *supports**(h_m, d), and *explains**(d, h_m).

Abductive inferences are uncertain rather than logically correct inferences that provide *evidence* for the hypotheses rather than conclusively proving them. The key source of uncertainty for any hypothesis is the possibility of alternative explanations for the data that supports the hypothesis. Even if *complete* support can be found for a hypothesis (i.e., there is a hypothesis or datum in *support-of*(h) that corresponds to each $S_i \in \mathcal{S}_T$), the hypothesis may still be uncertain as a result of competing, alternative explanations for *support-of**(h). Furthermore, because hypotheses are incrementally constructed, they may not have complete support at a given point in the processing even if complete support could be found in the available data (complete support might not be able to be found even if the hypothesis is correct due to masking, environmental disturbances, or sensor errors).

An *interpretation* of a data set is an explanation what caused all of the data. In general, an interpretation will be a *composite* of a set of hypotheses whose types are from a specified subset of the interpretation types (the *explanation corpus* [12]), each of which explains some subset of the data, and which together explain all of the data set. Formally, given data set D and interpretation I, if $h_i \in I$ then *support-of**$(h_i) \in D$ and $D =$ the union of *support-of**(h_i) forall $h_i inI$. A *solution* to an interpretation problem is an interpretation of the data that is judged best according to some criteria (we will return to this issue below).

In a centralized SI system, all of the sensor data is available to the single agent. In a distributed system, each agent has (direct) access to data from only a subset of the sensors, and each sensor is associated with a single agent. As a result, each agent monitors only a portion of the overall area of interest, and agents' local solutions must be combined in order to construct a global solution. Construction of a global solution may not be straightforward, however, because the local solutions are often not independent and may in fact be inconsistent because they are based on different incomplete subsets of the data. Agent solutions are interdependent whenever data (evidence) for hypotheses are spread among multiple agents or when agent areas of interest overlap as a result of overlapping sensor coverage.

In this paper we are interested in analyzing the quality of the global solutions that can be produced by FA/C distributed interpretation systems. To do this we need some standard for comparison—i.e., the "globally best solution." Usually one would like this to be the *most probable*

explanation (MPE) [12] given all of the globally available data.[4] The problem with defining the globally best solution to be the global MPE is that for most real-world SI problems it is impractical to compute the (true) MPE. For this reason and because our main interest is the effect of FA/C distributed problem solving, we have chosen to compare distributed SI solutions to those of an *equivalent* centralized system. By this we mean the solution that would be produced by a centralized (single-agent) system that has access to all of the data of the distributed system and that uses the same (possibly approximate) interpretation strategies as are used by the distributed agents.

The complexity of determining the true MPE can be understood by considering the differences between SI problems and the kinds of problems that are typically studied in research on abductive inference and probabilistic network inference (e.g., [12, 13]). For simplicity, we will refer to these problems as *diagnosis problems*. One key difference is that diagnosis problems are propositional while SI is not, in general. In other words, diagnosis problems have a fixed set of causes and possible findings, with fixed relations among them. The problem of identifying the MPE in such systems has been studied extensively (e.g., [12, 13]). By contrast, while SI problems have a fixed set of "top-level" cause *types* and a fixed set of data *types*, they can have an indeterminate number of instances of any of the types. For example, in a vehicle monitoring system, an unknown number of vehicles will have been responsible for the overall data set and each vehicle will produce a "track" of data points that can be of varying, unknown lengths.[5]

Because there can be an indeterminate number of instances of any top-level cause, a key problem for SI is what is known in the target tracking literature as the *data association problem* [1]: which target should data be associated with? The data association problem and the possibility of an indeterminate number of top-level causes lead to the problem of *correlation ambiguity* (it is ambiguous/uncertain which potential explanation hypothesis a data evidence should be associated with) and to a combinatorial explosion of possible explanations for a data set (potentially, every single piece of data could have a unique source instance as its cause).[6]

Other problems faced by SI systems, but not by most diagnosis problems relate to the nature of sensor data: evidence from sensor data is not conditionally independent even given a source/explanation hypothesis and there can be massive amounts of data resulting from multiple

[4]The MPE is not necessarily the *correct* interpretation—i.e., what actually produced the data. It is simply the most likely interpretation given the actual data and the model of how data can be produced by different events. The MPE can be incorrect when the characteristics of the data are unusual or when the system has a poor domain model. Also, the MPE may not be the "best" solution once one considers what will be done with the interpretation. For instance, failing to warn of a possible attacking aircraft has much more severe negative consequences (utility) than failing to identify a possible friendly one.

[5]Obviously the number of vehicles is usually not completely unknown because it can be bounded based on physical constraints or assumptions about the number vehicles that might appear. However, these constraints are not always very useful: large-scale military surveillance systems may have to deal with many hundreds of simultaneous vehicles.

[6]In general it is often impossible to *conclusively* rule out most of these interpretations, especially in applications where targets may be poorly sensed (i.e., often missed by the sensors).

passive sensors continuously operating in a noisy environment.[7] One way that we might sum up the differences is to say that in diagnosis problems the key source of complexity is the need to consider all combinations of causes for the evidence, while in SI problems it is the explosion of possible interpretations as large amounts of data are considered. Multiple cause combinations are often less of a problem for SI because data can often be assumed to have single causes, so most source hypotheses are alternatives.

Despite the differences between SI and diagnosis problems, it is useful to think of the networks of data and hypotheses that SI systems construct as being very similar to probabilistic belief networks. Key differences to keep in mind are: the SI hypothesis network will be incrementally constructed as data is interpreted, the complete network of possible interpretations may not be constructed, belief computations are often approximate, a noisy-OR model is usually not appropriate for modeling source/explanation interactions, and while networks may be drawn in such a way that data appears to be conditionally independent this is often not the case. Also, instead of random variables, in SI we talk about data and hypotheses, which can be complex, multi-attribute entities.

3 Approximate Interpretation Strategies

To deal with the characteristics discussed in the previous section, SI systems must be *constructive* [2, 5] and they must use approximate, satisficing strategies to determine solutions. It is not the point of this paper to investigate approximate strategies for SI, however to understand the significance of some of our results one must understand that that most real-world SI systems are approximate—even without the issues of distributed SI. We address this in our results by considering the effect of approximate interpretation strategies as well as the effect of different coordination strategies. Interpretation strategies are specified in general terms, since the particular strategies employed will be largely domain-specific. We describe alternative strategies via assumptions about the characteristics of the interpretations that result from the strategies that agents use for their local data. In the remainder of this section, we will give some sense of the kinds of approximation techniques that SI systems can use and their general effects. The distributed SI model we describe in the next section is based on the RESUN/DRESUN architecture, which provides great flexibility for implementing approximate interpretation strategies [2, 3].

There are five basic approximation techniques that can be used by SI systems: process only part of the available data, construct only some of the possible interpretation hypotheses for the processed data, compute approximate belief ratings (conditional probabilities) for the hypotheses,[8] consider only some of the possible composite interpretations (hypothesis combinations), and use

[7] The probabilistic belief network community is just starting to become aware of the data association problem and the issues it raises that have not been addressed in the work on diagnosis problems—e.g., [11].

[8] We will use the term "belief" to mean the degree-of-belief accorded to a hypothesis or interpretation. This will be either the conditional probability or approximate conditional probability of the object.

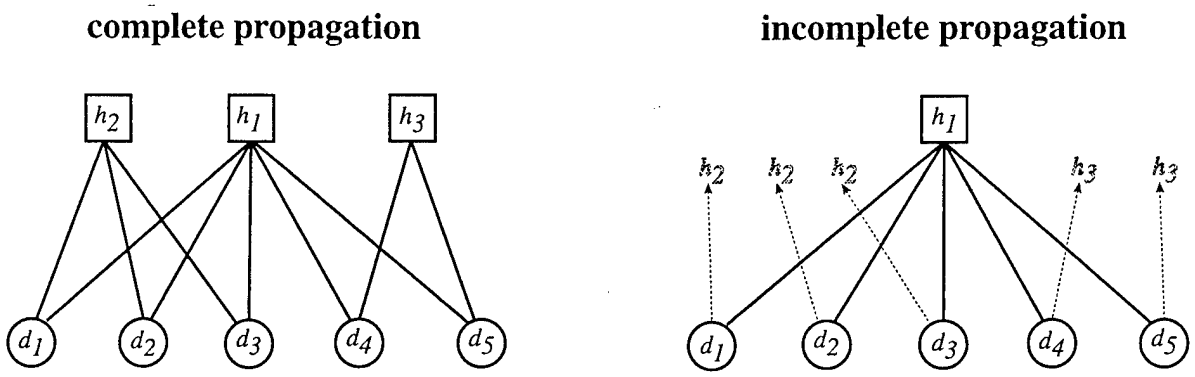


Figure 1: An example of approximate interpretation due to incomplete evidence propagation. In the complete propagation case, the system not only has created the most probable explanation, h_1, it also has created the alternative explanations, h_2 and h_3 (using the most complete support possible). This allows the system to determine the conditional probability of h_1 given the available data $\{d_1, \ldots, d_5\}$. In the incomplete propagation case, the alternative explanations for h_1 have not been created. This means that the belief computed for h_1 is only an approximation of the true conditional probability since the likelihood of the alternative explanations has been only approximately considered. h_1 is still uncertain, though, because the possibility of alternative type 2 and 3 explanations for each piece of supporting data is known, as are the a priori likelihoods of these explanations. (Note that for simplicitly, the figure does not show the numerous incompletely supported versions of the hypotheses that also might have been created in the case of complete propagation.)

some method other than maximum joint probability to select the solution interpretation. For example, the RESUN SI framework allows the use of heuristic knowledge to control the data that is processed and the hypotheses that are constructed, and it allows solutions to be assembled from hypotheses whose belief ratings simply surpass some *acceptance threshold.*

Obviously, these techniques are not independent. If a system does not process all of the data then it cannot in general create all possible interpretations of the complete data set. If a system does not create every possible interpretation of its data (i.e., every possible hypothesis h such that *explains**(D, h)), this not only limits the interpretations that can be considered, it also results in hypothesis belief ratings being only approximations of the true conditional probabilities of the hypotheses. This is because incomplete hypothesis construction results in incomplete propagation of the effects of evidence (we are using "evidence propagation" in the same basic sense that [12] refers to "belief propagation"). Figure 1 provides an example of this situation.

The bottom line is that these approaches will result in SI solutions that are only approximations of the MPE: they may be incomplete or they may not be the most probable composite interpretation. In the following sections we will consider both exact and approximate interpretation strategies being applied by agents to process their locally available data. We will use the name *exact local interpretation strategy* to refer to a strategy in which an agent's local solution (ignoring global interactions) would be the true MPE of the complete local data set. The term *approximate local interpretation strategy* will be used to refer to any strategy that is not exact. We will be more precise about the type and effect of approximate local strategies only as necessary.

One issue that we have not yet discussed, but which is of great importance with many approximate interpretation and coordination strategies is the *termination problem*: has the system done enough work to get a satisfactory solution? For SI this means has the system processed enough data, performed sufficient evidence propagation, and evaluated enough interpretations to have a reasonably good solution? We will return to the issue of termination criteria in our discussion of coordination strategies and in the results.

4 The DRESUN Distributed SI Model

In an FA/C system, there must be some mechanism to drive interactions among the agents so that incorrect and inconsistent local solutions can be detected and dealt with. Ideally, this would be accomplished with a mechanism that allowed agents to understand where there are constraints among their subproblems, so that information interchange could be highly directed. DRESUN [2, 3, 4] provides this capability, and it will form the basis for our model of the capabilities of an FA/C agent.

DRESUN agents create symbolic *source of uncertainty* statements (SOUs) to represent all of the reasons why their hypotheses are uncertain based on their local evidence. Whenever it is determined that a hypothesis can obtain evidence from another agent—i.e., whenever a subproblem interaction (constraint) is detected—DRESUN agents create *global consistency SOUs* (GSOUs) to represent the potential interactions. Each GSOU denotes that another agent might be able to provide additional evidence that would corroborate or contradict the associated local agent hypothesis.[9] GSOUs are viewed as sources of uncertainty about the correctness of an agent's local solution because until a GSOU is "resolved," it is uncertain whether the associated hypothesis is globally consistent. In this way, the basic DRESUN problem-solving goal of resolving interpretation uncertainty drives agents to communicate and produce solutions that are *globally consistent.*

To see that it is possible to identify all possible subproblem interactions, consider that in SI there are just three classes of evidential interactions between a local interpretation hypothesis and an external agent: the external agent might have duplicate support evidence due to overlapping sensor coverage, it might have support evidence for the hypothesis, or it might have explanation evidence. All instances of these situations can be easily detected given a local hypothesis, the SI domain model, and knowledge of the organization of agent interest areas. DRESUN uses three GSOUs to denote instances of these global interactions: *consistent-overlapping-model*, *consistent-global-extension*, and

[9]The GSOUs sometimes denote that an agent (or group of agents) is a *definite* source of evidence and sometimes only that an agent (or agents) is a *possible* source of evidence. Uncertainty arises from things like only partial coverage by an agent of the region from which evidence for a hypothesis might originate—e.g., the possible locations for the source vehicle for a ghost track might include area not completely monitored any agent.

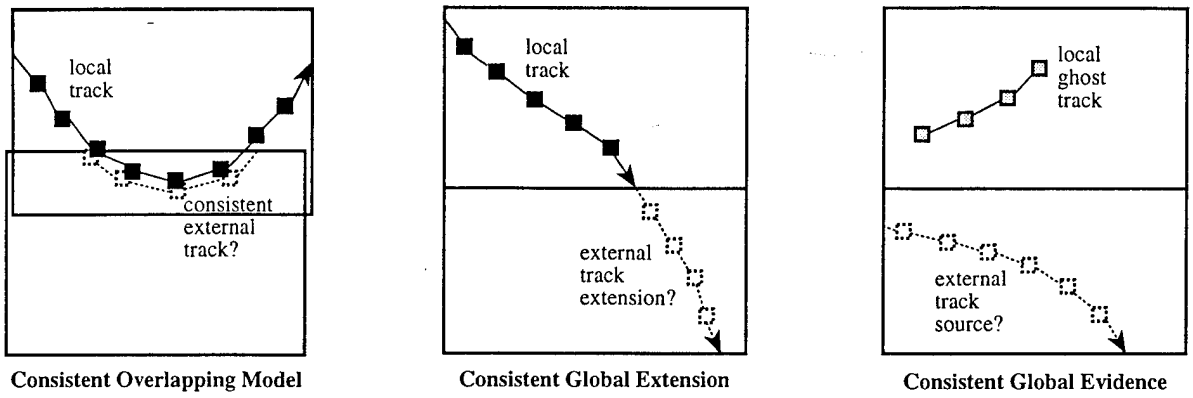


Figure 2: Examples of global consistency SOUs for vehicle monitoring.

consistent-global-evidence. Note that because of the special role of "continuous" support evidence in SI (a key type of support evidence that is not conditionally independent), a special GSOU is used to represent these support interactions while "individual" evidence interactions are represented with another GSOU (that is used for both support and explanation evidence). Figure 2 shows examples of vehicle monitoring situations involving each of these GSOUs.

To understand the effect of approximate local problem solving (interpretation) strategies on global solution quality, it is important to be clear that in DRESUN GSOUs are created based only on the interpretation hypotheses that are constructed. In other words, they do not result from direct analysis of the data.[10] Thus, approximate local interpretation strategies that do not construct all possible interpretations may result in a DRESUN system failing to detect the existence of potential global evidential interactions (which might affect the solutions that are chosen).

5 GSOU Resolution Strategies

While DRESUN agents have a representation of all inter-agent interactions (for the set of locally created hypotheses), we must now consider what should be done with this information in order to develop global solutions. In other words, what does *resolving* a GSOU mean, what are appropriate strategies for resolving the GSOUs, and how do these strategies affect the quality of the global solutions? Basically, resolution of a GSOU involves exchanging information between the associated agents so as to propagate the effects of each agent's local evidence between the agents. An example of the resolution of a GSOU is shown in Figure 3 (the figure will be discussed further below). Resolution of a GSOU is analogous to local (intra-agent) evidence propagation discussed in Section 3 in

[10]While interactions could be postulated directly from the data, this would result in large numbers of highly uncertain possible interactions. For example, in a large-scale military surveillance system it is possible for many targets over the entire area being monitored to be acting in concert, as in some attack scenario. As a result, a priori, there is some (non-zero) probability of *any* two pieces of data being interrelated in such a system.

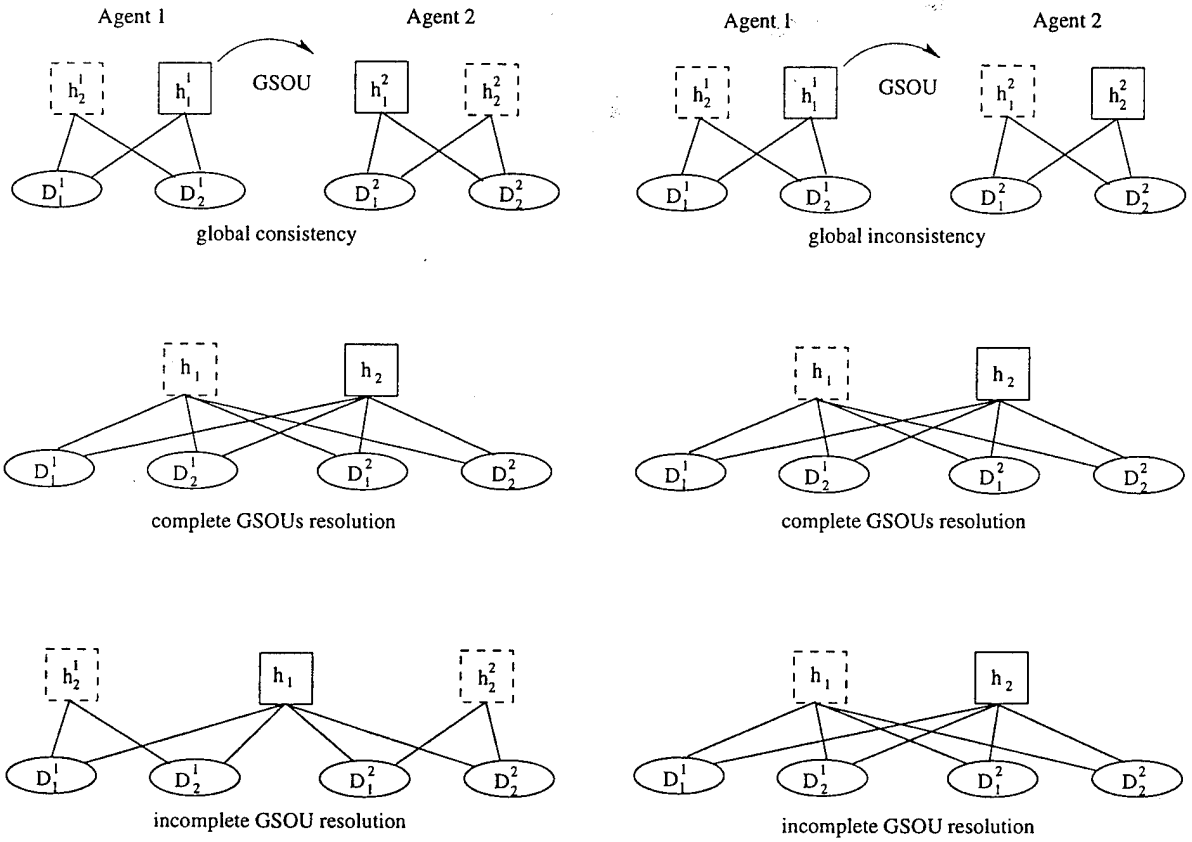


Figure 3: Examples of resolving a global consistency SOU (GSOU).

connection with the local interpretation strategies. As in local propagation, resolution of a GSOU may not be a one shot process, but may require periodic propagation as part of an incremental interpretation process. Thus, conceptually we can think of the GSOU resolution process as establishing evidential links among the associated agents' hypothesis networks (analogous to the links in a belief network).[11]

As with the local interpretation process, there are a range of GSOU resolution strategies that may be used. These strategies differ in how completely and accurately global interactions are considered, based on different choices about which global SOUs to pursue and how completely to assess the evidential effects of external agent data. In considering what are appropriate GSOU resolution strategies, we return to the issue of termination criteria. A key criterion that we wish to impose on FA/C agents is that their local solutions are "consistent" at termination. This allows the local solutions to be merged into a global solution without the need for additional problem solving. Our definition of global consistency of local solutions is an evidential one: solutions are

[11]Since inter-agent evidence propagation requires inter-agent communication, these links are only conceptual. In DRESUN, when evidence has been communicated to an external agent it is tagged. These tags alert the agent to communicate later modifications.

consistent if their hypotheses are pairwise identical, independent, or corroborative. Local solutions are inconsistent when any of their hypotheses are contradictory—i.e., have a negative evidential relationship.

The most comprehensive resolution strategy is for *all* global SOUs to be *completely* resolved—i.e., for all GSOUs in all agents to be resolved using the entire set of possible interpretation hypotheses constructed by the relevant external agents and creating all possible interpretation hypotheses from the joint interpretations (and their supporting data/evidence). We will refer to this as the *complete GSOUs resolution strategy.* An approximate coordination strategy that meets the global consistency criterion is for each agent to resolve only those GSOUs that are associated with its local solution ("best" interpretation) and to pursue evidence propagation using only the local solution hypotheses of the relevant external agents. We will refer to this as the *local solutions GSOUs resolution strategy.*

As we noted earlier, the overall effect of these strategies is dependent on the local interpretation strategy being used by the agents, since resolution is based on the interpretation hypotheses actually constructed by the agents.[12] In fact, there are other reasons why we cannot consider the effects of the global strategies in isolation from the local interpretation strategies. First, we must consider how the GSOUs resolution strategy interacts with the local interpretation strategy when the propagation of evidence among the agents results in an agent changing its belief ratings and/or creating new hypotheses. Our assumption will be that such changes trigger the agent to revise its solution, using the same criteria that it had been using for its local interpretation strategy. Thus, an agent using the *exact local interpretation strategy* will revise its solution to be the true MPE given its previous evidence and its newly acquired evidence. This is a reasonable assumption since SI systems often must be able to incrementally develop their solutions when they are operating continuously or providing users with current interpretations in real-time.

The second interaction issue, which is raised by the need to update the local solutions, is what information does the local interpretation strategy require be propagated between the agents. The local interpretation strategy can have a major impact on the amount of information that must be transferred among the agents to "resolve GSOUs" in the sense needed to meet the global consistency criteria. For example, approximate local strategies may simply require that global resolution provide sufficient information to compute the revised hypothesis beliefs that result from any interactions. By contrast, the *exact local interpretation strategy* has to consider the joint probabilities of the possible interpretations since it is to compute the MPE. In this case, if we are

[12]Thus, a more comprehensive strategy for resolving GSOUs would be one that considers not just the possible interpretations created by the external agents but also any of their data that could serve as evidence for the local hypotheses. We will not consider this alternative—it completely confuses the effects of the local and global strategies because the global strategy can effectively override the local strategy.

to truly have *complete* resolution of the GSOUs then enough information must be transferred to allow each agent to determine "their part" of the MPE given the joint evidence of the agents. When the MPE can be computed using a local computation, message passing scheme (see the discussion of belief revision in [12]) this is not a major issue. However, if the agents require access to all of the possible component hypotheses of the possible interpretations then substantial additional evidence may have to be communicated (beyond that necessary to compute revised hypothesis beliefs).

Figure 3 shows examples of resolving a GSOU for both the *exact local interpretation strategy* and the *local solutions GSOUs resolution strategy.* One thing to note from this figure is the effect that global solution consistency/inconsistency has on the amount of inter-agent propagation that occurs using the two strategies. As the figure shows, the *exact local interpretation strategy* performs the same level of inter-agent evidence propagation regardless of whether the local solutions are consistent or not. By constrast, the *local solutions GSOUs resolution strategy* is an approximate strategy because it does not completely propagate the effects of the agents evidence when the local solutions are consistent. However, it performs exactly the same level of propagation as the *exact local interpretation strategy* when the local solutions are inconsistent.

6 Notation

This section defines the notation that will used in presenting the results and their proofs that appear in the following section. For the centralized (single-agent) case, we will use the following notation:

- $\mathcal{D}$ is the complete data set available to the system.
- D_i denotes some subset of $\mathcal{D}$ ($D_i \subseteq \mathcal{D}$).
- C_i denotes an interpretation *context*—i.e., a particular stage of processing. Each context refers to a specific data subset D_i that has been processed and a particular set of hypotheses and evidential links that have been created.
- $BEL(h, C_i)$ denotes the true degree-of-belief in hypothesis h in context C_i—i.e., $P(h \mid D_i)$, where D_i is the data subset associated with context C_i. This may not be the belief rating actually associated with h—see $\widehat{BEL}(h, C_i)$ below. Also, in general $BEL(h, D_i) \neq BEL(h, \mathcal{D})$ when $D_i \subset \mathcal{D}$, and because interpretation domains are not typically monotonic $BEL(h, D_i)$ may be less than or greater than $BEL(h, \mathcal{D})$. When the interpretation context is clear, we will simply use $BEL(h)$.
- $\widehat{BEL}(h, C_i)$ is the belief rating that is actually associated with h given the interpretation strategy being used. In general $\widehat{BEL}(h, C_i) \neq BEL(h, C_i)$ unless the *exact local interpretation strategy* is being used. As above, $\widehat{BEL}(h, C_i)$ will be written as $\widehat{BEL}(h)$ when the context is clear.
- $\mathcal{I}(\mathcal{D})$ is the true MPE of the complete data set $\mathcal{D}$.
- $\mathcal{I}(C_i)$ is the true MPE of the data subset associated with context C_i. This may not be the solution that is selected by the interpretation strategy—see $\hat{\mathcal{I}}(C_i)$ below.

- $\hat{\mathcal{I}}(C_i)$ is the "best interpretation" solution of the data subset associated with context C_i given the local interpretation strategy being used. We do not specify how the solution interpretation is selected, but it need not be the MPE given the approximate processing represented by C_i (the solution may for instance have been selected using acceptance thresholds).

For the distributed, multi-agent case, modifications must be made to account for the distribution of data and hypotheses among multiple agents. In particular, the concept of an interpretation context must be extended for the distributed case to deal with there being multiple contexts related to each agent and with evidence communication/propagation among the agents as part of GSOUs resolution. The following notation will be used for the distributed case:

- $\mathcal{A}$ is the set of agents $\{A_1, A_2, \ldots\}$, with their interest area specifications.
- $\mathcal{D}^i$ is the complete data set available (directly) to only agent A_i—i.e., the complete data set that is available from agent A_i's own sensors.
- $\mathcal{D}^G$ refers to the complete, globally available data set—i.e., the combined data from all of the agents, $\bigcup_{A_i \in \mathcal{A}} \mathcal{D}^i$. $\mathcal{D}^G = \mathcal{D}$ for an equivalent centralized system.
- D^i_j denotes some subset of agent A_i's local data ($D^i_j \subseteq \mathcal{D}^i$).
- $D^{j \rightarrow i}_k$ refers to external data from agent A_j, D^j_k , that is available to agent A_i because it has been communicated from A_j as part of GSOUs resolution. Note that the evidential effects of this data may only partly be known to A_i depending on the local interpretation strategy being used by A_j.
- $D^{G \rightarrow i}_k$ refers to external data from all the other agents, $\bigcup_{A_l \neq A_i} D^l_k$, that is available to agent A_i because it has been communicated by these agents as part of GSOUs resolution.
- C^i_j denotes an interpretation context for agent A_i. As with C_i above, each context refers to a specific subset of the local data D^i_j that has been processed and a particular set of hypotheses and evidential links that have been created. In addition, it also refers to any available external evidence that has been communicated from other agents in the process of resolving GSOUs, $D^{G \rightarrow i}_j$.
- C^G_j will be used to refer to a global context—i.e., $C^G_j \equiv \bigcup_{A_i \in \mathcal{A}} C^i_j$.
- $BEL^i(h, C_j)$ denotes the true degree-of-belief in hypothesis h in agent A_i's context C^i_j. It is just $P(h \mid D^i_j, D^{G \rightarrow i}_j)$. This may not be the belief rating that agent A_i actually associates with h—see $\widehat{BEL}^i(h, C_j)$ below. When the interpretation context is clear, we will simply use $BEL^i(h)$.
- $\widehat{BEL}^i(h, C_j)$ is the belief rating that agent A_i actually associates with h in context C^i_j given the interpretation strategy being used. $\widehat{BEL}^i(h, C_j) \neq BEL^i(h, C_j)$ unless the agents are using the *exact local interpretation strategy*. As above, $\widehat{BEL}^i(h, C_j)$ will be written as $\widehat{BEL}^i(h)$ when the context is clear.
- $\mathcal{I}^i(C_j)$ is the MPE for the data subset associated with context C^i_j.
- $\hat{\mathcal{I}}^i(C_j)$ is agent A_i's "best interpretation" solution of the data subset associated with context C^i_j—given the local interpretation and global resolution strategies being used. As in the

centralized case, we do not specify how the solution interpretation is selected but it need not be the MPE given the approximate processing represented by C_j^i.

- $\hat{\mathcal{I}}^G(C_j)$ denotes the combined, global "best interpretation" solution of the data subset associated with the global context C_j^G. That is, $\hat{\mathcal{I}}^G(C_j) = \bigcup_{A_i \in \mathcal{A}} \hat{\mathcal{I}}^i(C_j)$. $\hat{\mathcal{I}}^G(C_j)$ is defined only when all the $\hat{\mathcal{I}}^i(C_j)$ are consistent, as in a final context that satisfies the consistency termination criteria.

7 Solution Quality Results

In this section we will examine the quality of solutions that can be produced using an FA/C approach, relative to those that could be produced using an equivalent centralized approach. To rephrase the main question we are interested in addressing: what is the relationship between $\hat{\mathcal{I}}^G(C_{f_d}^G)$ and $\hat{\mathcal{I}}(C_{f_c})$, where $C_{f_d}^G$ and C_{f_c} represent possible final contexts (contexts that satisfy the termination criteria) for the distributed and centralized cases, respectively?

7.1 Theorem 1

In this section we will examine the quality of FA/C global solutions assuming that the centralized system and the distributed agents use the *exact local interpretation strategy* to process their local data and that the distributed agents use the *complete GSOUs resolution strategy* (the strategies are described in Sections 3 and 5).

Theorem 1: *Given a centralized system that uses the* exact local interpretation strategy, *a set of agents $\mathcal{A}$ each of which uses the* exact local interpretation strategy *and the* complete GSOUs resolution strategy, *$\forall \mathcal{D}$ and $\forall \{\mathcal{D}^i\}$ with $\mathcal{D}^G = \mathcal{D}$ (all possible divisions of the data among the agents), $\hat{\mathcal{I}}^G(C_{f_d}^G) = \hat{\mathcal{I}}(C_{f_c}) = \mathcal{I}(\mathcal{D})$.*[13] In other words, when agents use the *exact local interpretation strategy* and the *complete GOUs resolution strategy*, the FA/C distributed system produces the same exact interpretation as the centralized system, which is the true MPE of all the globally available data.

Proof: Clearly $\hat{\mathcal{I}}(C_{f_c}) = \mathcal{I}(\mathcal{D})$ by the definition of the *exact local interpretation strategy* and the fact that the centralized system (agent) has access to the complete data set $\mathcal{D}$. To show that $\hat{\mathcal{I}}^G(C_{f_d}^G) = \mathcal{I}(\mathcal{D})$ we must show that if $h \in \hat{\mathcal{I}}^G(C_{f_d}^G)$ then $h \in \mathcal{I}(\mathcal{D})$ and if $h \in \mathcal{I}(\mathcal{D})$ then $h \in \hat{\mathcal{I}}^G(C_{f_d}^G)$. Suppose that $h \in \hat{\mathcal{I}}^G(C_{f_d}^G)$. This means that there is at least one $A_i \in \mathcal{A}$ such that $h \in \hat{\mathcal{I}}^i(C_{f_d}^i)$. We specified that when using these two strategies in conjunction, enough evidential

[13] Note that we can refer to the global interpretation $\hat{\mathcal{I}}^G(C_{f_d}^G)$ rather than just individual agent interpretations $\hat{\mathcal{I}}^i(C_{f_d}^G)$ since in the final context $C_{f_d}^G$ individual agent solutions must be consistent (mergeable).

information must be transferred among the agents so that the local agent solutions are not only consistent, but together represent the MPE given the evidence based on their local hypotheses. If we assume that every agent creates every possible interpretation of its data as part of the *exact local interpretation strategy*, then this will mean that every possible inter-agent evidential relation will be represented by some GSOU and so will be resolved. Thus, in a final context where all GSOUs have been completely resolved and exact local interpretations have been computed, h must be in $\mathcal{I}(\mathcal{D})$, the MPE given the globally available data. Now, suppose that $h \in \mathcal{I}(\mathcal{D})$. If we again assume that agents are creating every possible interpretation hypothesis as part of the *exact local interpretation strategy*, h will have been created by at least one agent A_i. There are two cases to consider: 1.) $BEL^i(h, \mathcal{D}^i) = BEL(h, \mathcal{D})$ and 2.) $BEL^i(h, \mathcal{D}^i) \neq BEL(h, \mathcal{D})$. In the first case, the belief in h is independent of the data in all external agents. This means that A_i can locally assess the belief in h and whether it is part of the MPE without any resolution of GSOUs. Thus, if $h \in \mathcal{I}(\mathcal{D})$ then $h \in \hat{\mathcal{I}}^i(C^i_{f_d})$ and so $h \in \hat{\mathcal{I}}^G(C^G_{f_d})$ (by the global consistency criteria). In the second case, A_i cannot locally assess the belief in h. However, if we assume that all possible interpretation hypotheses are created and all GSOUs are completely resolved, then agent A_i must have the information necessary to assess both the belief in h and whether it is in the MPE. Thus again, if $h \in \mathcal{I}(\mathcal{D})$ then $h \in \hat{\mathcal{I}}^i(C^i_{f_d})$ and so $h \in \hat{\mathcal{I}}^G(C^G_{f_d})$ (by the global consistency criteria).

This result is what we would expect given the definitions of the *exact local interpretation strategy* and the *complete GOUs resolution strategy*, and given the assumptions inherent in the DRESUN model (that the GSOUs represent all possible interactions for the set of created hypotheses). However, it serves to formally establish the theoretical capabilities of a DRESUN-based FA/C distributed SI system. Other FA/C distributed SI architectures have not had this property because they have lacked a representation of all the inter-agent interactions and a mechanism for controlling their resolution. This result shows that an incremental, distributed approach to SI need not produce poorer quality solutions than a centralized system.[14] Furthermore, the result is not trivially obvious. We had to make the assumption that agents create all possible interpretation hypotheses and we also had to be very careful in how we defined the interactions between the local interpretation and GSOUs resolution strategies for the result to hold.

Unfortunately, the practical implications of this result are very limited since few SI systems (centralized or distributed) can afford to do exact local interpretation and complete GSOUs resolution. The only time such a situation would be practical would be if enough agents are used so that each agent's local processing requirements are adequate to completely and exactly process its

[14] We assumed that both systems had access to exactly the same data. This is reasonable if only passive sensors are being used or if the SI is not being done in real-time, but is much less reasonable once active sensors can be used in real-time SI.

local data and if the global interactions are limited in both number and scope. In other words, if few GSOUs are produced and those that are can be resolved with limited processing. This latter point is important, since even if there are few GSOUs, it may be very costly to completely resolve every one if they can each potentially interact with large numbers of interpretation hypotheses in the associated external agent(s).

It is easy to show that the use of approximate local interpretation strategies or an incomplete GSOUs resolution strategy results in the systems failing to be guaranteed of producing the MPE. However, if we accept that even centralized SI systems will have to settle for sub-optimal solutions, then what we are really interested in determining is whether distributed systems can produce solutions that are equivalent with (or at least sufficiently close to) those produced by approximate centralized systems, and whether they can do it efficiently.[15] It certainly is possible to come up with approximate local interpretation strategies for an approximate FA/C-based SI system to produce solutions that are equivalent to those of an approximate centralized system. However, this depends on the details of the approximate interpretation strategy since there are only certain approximation strategies whose effects can be duplicated by a GSOUs resolution strategy (the agents in the distributed system will still have limited views of the globally avaiable raw data, for instance). In the remaining results subsections, we will begin to consider approximate interpretation and coordination strategies.

7.2 Theorem 2

We are not yet at the point where we can formally define and analyze the effects of arbitrary interpretation and coordination strategies. However, in this section we will undertake some evaluation of the effects of the *local solutions GSOUs resolution strategy*. This is an important strategy because it can be very efficient when local agent solutions are consistent. In addition, the developers of the FA/C paradigm clearly saw this as a useful coordination strategy. For example, [8] refers to *consistency checking* of the tentative local solutions with results received from other nodes as "an important part of the FA/C approach."

Theorem 2: *Given a centralized system that uses an arbitrary (possibly approximate) local interpretation strategy and a set of agents $\mathcal{A}$ each of which uses this same local interpretation strategy and the* local solutions GSOUs resolution strategy, $\exists \mathcal{D}$ *and* $\exists \{\mathcal{D}^i\}$ *with* $\mathcal{D}^G = \mathcal{D}$ *(some*

[15] To clarify this discussion, we should point out that a distributed SI system does not necessarily do more "work" than the centralized system does to get the MPE solution. Completely resolving the GSOUs results in the distributed system performing precisely the same evidential propagation as would have to be performed by a centralized system. Of course, doing this propagation via inter-agent communication will be more costly both in terms of processing and elapsed time than it would be in a centralized system—that is the issue. The basic complexity of SI is determined by the interrelatedness/independence of the possible data interpretation hypotheses, though.

division of the data among the agents), such that $\hat{\mathcal{I}}^G(C^G_{f_d}) \neq \hat{\mathcal{I}}(C_{f_c})$ *and* $\hat{\mathcal{I}}^G(C^G_{f_d}) \neq \mathcal{I}(\mathcal{D})$. In other words, the *local solutions GSOUs resolution strategy* does not guarantee that the FA/C distributed system produces the same exact interpretation as the centralized system or the true MPE of all the globally available data.

Proof: Regardless of the local interpretation strategy being used, there must be some data and some distribution of that data so that there is some hypothesis h such that $h \in \hat{\mathcal{I}}^i(\mathcal{D}^i)$ and $h \in \hat{\mathcal{I}}^j(\mathcal{D}^j)$, but not $h \in \hat{\mathcal{I}}(C_{f_c})$. The only case in which this would not be true would be a domain in which there was complete solution monotonicity—i.e., $\forall D_i \subset \mathcal{D}$, if $h \in \mathcal{I}(D_i)$ then $h \in \mathcal{I}(\mathcal{D})$.

For example, it is entirely possible to have $P(H_a \mid D_1) > P(H_b \mid D_1)$ and $P(H_a \mid D_2) > P(H_b \mid D_2)$, but $P(H_a \mid D_1, D_2) < P(H_b \mid D_1, D_2)$ (where H_a and H_b are competing hypotheses and D_1 and D_2 are data sets in different agents). In other words, the solution that is locally most likely in both agents (H_a) may not be the globally most likely solution—even though the local solutions are consistent. This is simply a consequence of the nonmonotonicity inherent in reasoning based on incomplete and uncertain information.

7.3 Approximate Monotonicity

Because "consistency" of local solutions does not in general guarantee that the merged, global solution is the best solution, the *local solutions GSOUs resolution strategy* may not produce solutions equivalent to an approximate centralized system, let alone the MPE. This is unfortunate since this coordination strategy can be very efficient. It also conflicts with the intuitions of ourselves and other other FA/C researchers that this is a reasonable approximate FA/C coordination strategy.

We believe that the reason for this apparent contradiction is that many real world domains have characteristics that allow the this strategy to work quite well. For example, in vehicle monitoring, a considerable amount of evidence must typically be acquired to achieve a vehicle track hypothesis with high belief. While additional evidence can negatively affect the belief in such a track, it will take substantial additional evidence to have a major effect on the belief. In other words, while belief is nonmonotonic with increasing evidence, in some situations the effects of additional evidence are not totally unpredictable.

What this observation means for the "local solutions strategy" is that if a track hypothesis is part of a local solution, has "fairly high" belief, and is globally consistent, then it is pretty likely that the track is correct. We will refer to this characteristic as *approximate monotonicity.* We have formalized this notion in terms of probability distributions for the correct global probability of a hypothesis in terms of its belief based on local agent data only. A key focus of our future research

will be to use this concept to design a coordination strategy whose effects are predictable in terms of the probability of its global solution being optimal.

7.4 Theorem 3

In this section and the next, we will consider the use of another approximate GSOUs resolution strategy, one that is *incremental* in the sense that the completeness with which the GSOUs are resolved is not prespecified but depends on the situation. The termination criteria will now be phrased in terms of whether a solution is *acceptable* or not. We define the predicate $acceptable(\mathcal{I}(C))$ to mean that $\forall h \in \mathcal{I}(C), \widehat{BEL}(h) \geq AT$, where AT is the *acceptance threshold* $(0 < AT \leq 1)$. Based on this definition we define the *incremental GSOUs resolution strategy* as: each agent will resolve the GSOUs as in the *local solutions GSOUs resolution strategy*, but if its resulting local solution is not *acceptable* it must continue to resolve global SOUs associated with the hypotheses in its solution until the solution is judged acceptable or all the relevant global SOUs have been resolved (i.e., as when using the *complete GSOUs resolution strategy*).

In this section we will assume that the agents use the *exact local interpretation strategy.* For the centralized case, this means that the final context C_{f_c} refers to the complete data set $\mathcal{D}$, and that $\hat{\mathcal{I}}(C_{f_c}) = \mathcal{I}(\mathcal{D})$. Because we are now considering a satisficing approach to solution selection, instead of comparing the centralized and distributed solutions based on whether they are idential, we will judge them based on whether they are *acceptable.*

Theorem 3: *Given the* exact local interpretation strategy *and the* incremental GSOUs resolution strategy, $\forall \mathcal{D}$ *if* $acceptable(\hat{\mathcal{I}}(C_{f_c}))$ *then* $acceptable(\hat{\mathcal{I}}^G(C^G_{f_d}))$, *where* C_{f_c} *and* $C^G_{f_d}$ *are final contexts that meet the specified centralized and distributed termination criteria for the* $\mathcal{D}$, *respectively.* In other words, if a centralized SI system is able to produce an acceptable solution for a data set then so should a distributed DRESUN system using the specified strategies.

Proof: Assume that the theorem is false. This means that even though $acceptable(\hat{\mathcal{I}}(C_{f_c}))$, $\exists h \in \hat{\mathcal{I}}^G(C^G_{f_d}))$ such that $\widehat{BEL}^G(h, C^G_{f_d}) < AT$. Now, because $acceptable(\hat{\mathcal{I}}(C_{f_c}))$, $\exists \{h_j\} \subset \hat{\mathcal{I}}(C_{f_c})$ such that $explains^*(support\text{-}of^*(h), \{h_j\})$ and $\forall h_j \in \{h_j\}, BEL(h_j) \geq AT$. In other words, the centralized system has one or more hypotheses that are part of its solution, that have acceptable belief ratings, and that explain the data underlying the unacceptable hypothesis h in the global interpretation. Since these hypotheses are alternative explanations for $support\text{-}of^*(h)$, each of these hypotheses would also have been created by at least one of the agents that produced h.[16] So, for

[16] Actually, the exact same hypothesis may not have been produced by a single agent since each of the agents may have only a portion of the data necessary to produce the version of the hypothesis with the most *complete* support possible. This issue arises because interpretation problems are not propositional, so as additional evidence is generated for a hypothesis the evidence both modifies the hypothesis' belief and refines the values of its attributes—creating

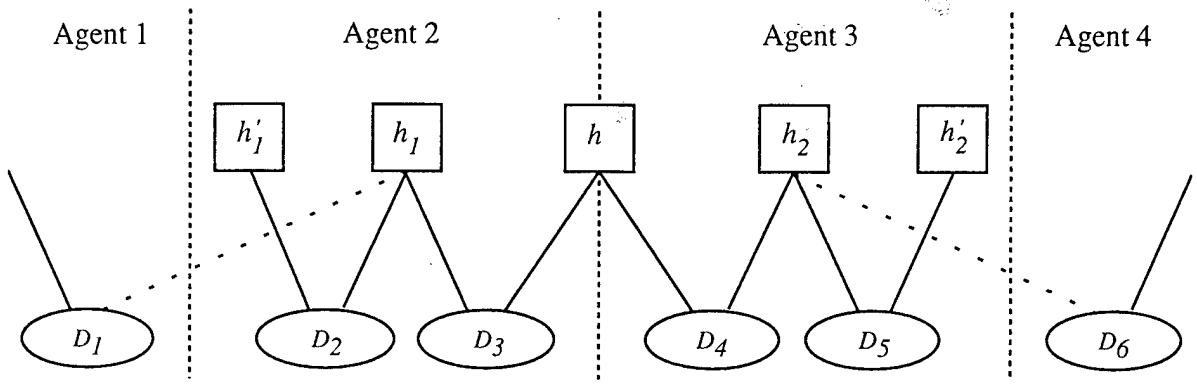


Figure 4: An example of the effect of partial resolution of the global SOUs.
Here, resolution of a GSOU btween agents 2 and 3 produced the hypothesis h, which resulted in $\{h, h_1', h_2'\}$ being judged as an acceptable explanation for agents 2 and 3. Because of this, the GSOUs associated with h_1 and h_2 were not resolved. However, resolution of these GSOUs could result in corroborating evidence being obtained from agents 1 and 4, which might make $\{h_1, h_2\}$ the best explanation.

each h_j there is a corresponding hypothesis h_j^k that would have been created by A_k. Now, why is $\widehat{BEL}^k(h_j^k) < AT$ when $BEL(h_j) \geq AT$? There are three possibilities: (1) there exists data in one or more agents other than A_k that would corroborate h_j^k, but this data was not propagated to h_j^k;[17] (2) there exists data in one or more other agents that would contradict one or more alternatives to h_j^k; (3) both of the above may be true. Consider case 1: If there exists data in other agents that could support h_j^k, then A_k would have had GSOUs associated with h_j^k that identified the possible existence of this data (since GSOUs identify all interrelationships among the agents given the constructed hypotheses), but these SOUs were not resolved. This leads to a *contradiction* with our specified coordination strategy, however, since these SOUs should have been pursued when no acceptable explanation was found for the data common to h and h_j^k (remember that $h \in \hat{\mathcal{I}}^G(C_{f_d}^G)$ but $\widehat{BEL}^G(h, C_{f_d}^G) < AT$). Cases 2 and 3 lead to similar contradictions.

Thus, given the specified strategies if a centralized interpretation system can find an interpretation above a particular threshold then so can a DRESUN-based distributed interpretation system. This is a somewhat weak result, but it serves to establish a basic capability of the DRESUN architecture. Other FA/C distributed problem-solving architectures have not had this property because they have lacked a representation of the inter-agent constraints.

It is important to note that while we have shown that a distributed DRESUN system would find an "acceptable" solution, this interpretation may not be identical to that produced by the

new versions of the hypothesis (what [2] terms *extensions*). Thus, what we really mean is that a precursor, partial version of each alternative hypothesis would have been created by one or more of the agents. This is true because interpretation hypotheses can be created based on incomplete support, and our specified strategy was for each agent to create all the possible interpretations that are (even partially) supported by its own local data.

[17]The data cannot be in A_k since we have assumed complete intra-agent propagation.

centralized system and may not be the MPE of the global data set. Because an acceptable solution might be found without all the GSOUs having been resolved, there could have been incomplete evidence propagation among the agents. As a result, hypothesis belief ratings would be only approximations of the true conditional probabilities and this could lead to alternative explanations being (incorrectly) rated more highly—see Figure 4. This is the reason that the proof used $BEL(h)$ for the centralized system, but $\widehat{BEL}^G(h)$ for the distributed system. As Theorems 1 and 2 show, the only way to guarantee that a distributed system produces a solution that is *identical* to that of the centralized system (and is the MPE of the globally available data) is to require that all of the GSOUs associated with every hypothesis in each agent be completely resolved.

7.5 Theorem 4

In this section we will again examine the nature of global solutions when using the *incremental GSOUs resolution strategy.* However, unlike in Theorem 3, here we will assume that the agents use an *approximate local interpretation strategy.* This means that agents do not necessarily construct all possible interpretation hypotheses and so may compute only approximate belief ratings (conditional probabilities) for the hypotheses. Analyzing the solution quality given incomplete evidence propagation is complicated by the fact that a system's solutions can depend on the details of the strategies being used. We are not yet at the point where we can formally define and analyze the effects of different strategies. Thus, we will not specify a particular strategy for controlling evidence propagation here. Instead, we will simply assume that the same strategy is used in the single agent of the centralized case and in all of the agents of the distributed case. We will continue to assume that all of the data directly available to each agent is (at least partially) processed by that agent.

Theorem 4: *Given an* approximate local interpretation strategy *and the* incremental GSOUs resolution strategy, *it is not the case that,* $\forall \mathcal{D}$ *if* $acceptable(\hat{\mathcal{I}}(C_{f_c}))$ *then* $acceptable(\hat{\mathcal{I}}^G(C^G_{f_d}))$, *where* C_{f_c} *and* $C^G_{f_d}$ *are final contexts that meet the specified centralized and distributed termination criteria for the data set* $\mathcal{D}$, *respectively.* In other words, even if a centralized system is able to produce an acceptable solution for a data set, a distributed DRESUN system may not (even when the same local interpretation strategy is being used).

Proof: Assume that the theorem is false: $\forall \mathcal{D}$ $acceptable(\hat{\mathcal{I}}(C_{f_c}))$ implies $acceptable(\hat{\mathcal{I}}^G(C^G_{f_d}))$. It is easy to show the general existence of counter examples (the counter example is illustrated in Figure 5). Suppose that, the centralized system, which has a view of the complete data set $D_1 \cup D_2 \cup D_3 \cup D_4$, initially chooses to construct hypotheses h_1 and h_2. If this is an acceptable explanation ($\widehat{BEL}(h_1)$ and $\widehat{BEL}(h_2) \geq AT$) no further propagation may be done. In the distributed case, the agents might initially produce the hypotheses h_3 and h_4 (based on their own local data).

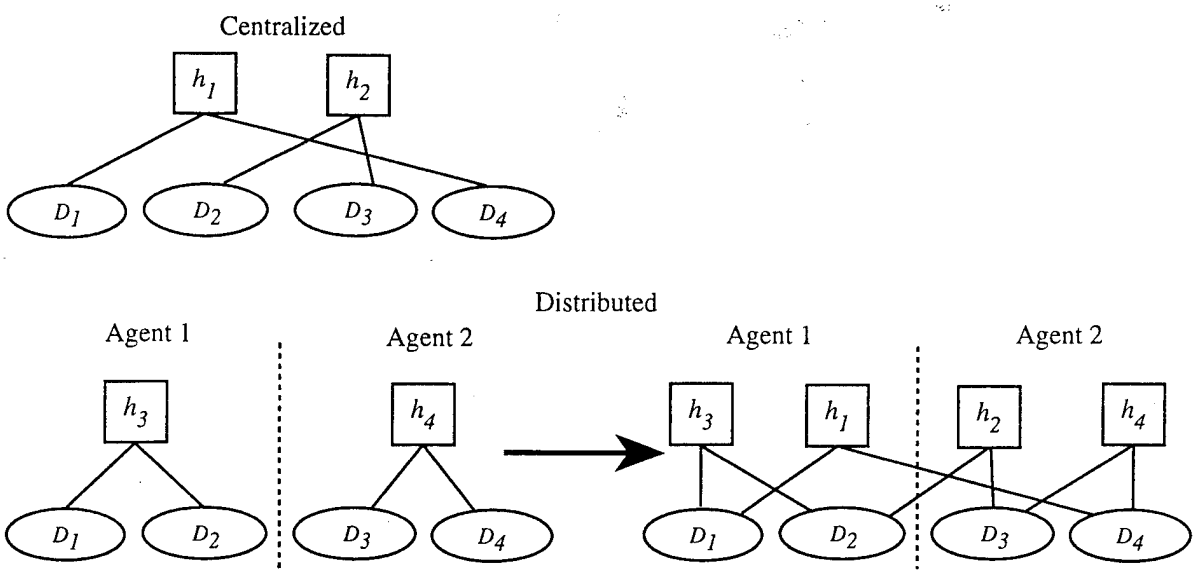


Figure 5: A counter example for the proof of theorem 4.

In general, there will certainly exist data scenarios for which this interpretation would not be acceptable. However, even if the agents communicate and construct h_1 and h_2, $\widehat{BEL}(h_1)$ and $\widehat{BEL}(h_2)$ may be different than the approximate values computed in the centralized case, and could in fact lead to this alternative interpretation also being judged unacceptable.

The reason that an acceptable centralized solution does not guarantee an acceptable distributed solution is that the division of data among the distributed agents may lead to the construction of a different set of hypotheses from those constructed by the centralized system. The construction of different hypotheses produces different belief approximations. This could cause the distributed system to produce a different acceptable solution or it could cause it to fail to find any acceptable solution. Even if the distributed system eventually constructs the same interpretation hypotheses as the centralized system, it may have done additional evidence propagation that causes the interpretation to be judged unacceptable. Furthermore, just because the distributed system may be driven to do more evidence propagation than the centralized system, this does not mean that its best solution is more likely to be the (true) global MPE (as long as its propagation is still incomplete).

7.6 Theorem 4 Bounds

Another way of stating theorem 4 is that when approximate GSOUs resolution strategies are used, there is merely some probability of a distributed system producing an acceptable solution when an acceptable solution is produced by an equivalent centralized system. This probability is a function of the interpretation and coordination strategies being used, the acceptance threshold,

the characteristics of the domain, and the organization of agent interest areas. Understanding the effects that these factors have on the probability of achieveing comparable global solutions is a major focus of our future research. As a first step, we have explored how to decompose this probability so that we could determine bounds on its components based on reasonable probabilistic characterizations of a control strategy and domain.

So what we want to determine is: $P(acceptable(\hat{\mathcal{I}}^G(C^G_{f_d})) \mid acceptable(\hat{\mathcal{I}}(C_{f_c})))$, where C_{f_c} and $C^G_{f_d}$ are final contexts that meet the specified centralized and distributed termination criteria for the $\mathcal{D}$, respectively. (For simplicity, we will use $\hat{\mathcal{I}}^G$ for $\hat{\mathcal{I}}^G(C^G_{f_d})$ and $\hat{\mathcal{I}}^C$ for $\hat{\mathcal{I}}(C_{f_c})$ from this point on.) One way to decompose the probability is:

$$
\begin{aligned}
&P(acceptable(\hat{\mathcal{I}}^G) \mid acceptable(\hat{\mathcal{I}}^C)) = \\
&\quad P(acceptable(\hat{\mathcal{I}}^G) \mid acceptable(\hat{\mathcal{I}}^C) \wedge \hat{\mathcal{I}}^G = \hat{\mathcal{I}}^C) * P(\hat{\mathcal{I}}^G = \hat{\mathcal{I}}^C \mid acceptable(\hat{\mathcal{I}}^C)) + \\
&\quad P(acceptable(\hat{\mathcal{I}}^G) \mid acceptable(\hat{\mathcal{I}}^C) \wedge \hat{\mathcal{I}}^G \neq \hat{\mathcal{I}}^C) * P(\hat{\mathcal{I}}^G \neq \hat{\mathcal{I}}^C \mid acceptable(\hat{\mathcal{I}}^C))
\end{aligned}
$$

Here, we have decomposed the situation based on whether the distributed system finds that the most likely interpretation is the same interpretation as was found by the centralized system or not (remember that though the distributed system may have produced the same interpretation as the centralized system it may have done different propagation, so its belief ratings for the hypotheses of the interpretation may be different from those of the centralized system). Based on this decomposition, it is possible to identify some bounds on the component probabilities.

Consider the first conditional probability term in the decomposition:

$$
\begin{aligned}
&P(acceptable(\hat{\mathcal{I}}^G) \mid acceptable(\hat{\mathcal{I}}^C) \wedge \hat{\mathcal{I}}^G = \hat{\mathcal{I}}^C) \geq \\
&\quad P(\forall h_i \in \hat{\mathcal{I}}^C, BEL(h_i) \geq AT \mid acceptable(\hat{\mathcal{I}}^C))
\end{aligned}
$$

Proof: To prove this, we need to show that $acceptable(\hat{\mathcal{I}}^C) \wedge (\forall h_i \in \hat{\mathcal{I}}^C, BEL(h_i) \geq AT)$ implies $acceptable(\hat{\mathcal{I}}^G)$ (when $\hat{\mathcal{I}}^G = \hat{\mathcal{I}}^C$). Assume that this is false, so that $acceptable(\hat{\mathcal{I}}^C) \wedge (\forall h_i \in \hat{\mathcal{I}}^C, BEL(h_i) \geq AT) \wedge \neg acceptable(\hat{\mathcal{I}}^G)$. Based on the specified propagation strategy, $\neg acceptable(\hat{\mathcal{I}}^G)$ would cause the distributed system to do further evidence propagation until either $acceptable(\hat{\mathcal{I}}^G)$—which would contradict the assumption—or complete propagation has been done for all hypotheses in $\hat{\mathcal{I}}^G$ that are not acceptable—which also leads to a contradiction, as we will show. The complete propagation situation means that $\forall h_i \in \hat{\mathcal{I}}^G$ such that $\widehat{BEL}^G(h_i) < AT, \widehat{BEL}^G(h_i) = BEL^G(h_i)$. However, this contradicts the premise assumptions that $\forall h_i \in \hat{\mathcal{I}}^C, BEL(h_i) \geq AT$ since $\hat{\mathcal{I}}^G = \hat{\mathcal{I}}^C$ here.

$P(\forall h_i \in \hat{I}^C, BEL(h_i) \geq AT \mid acceptable(\hat{I}^C))$ is the probability that the hypotheses in the centralized interpretation are "acceptable" given these hypotheses true belief ratings whenever the partial propagation strategy finds that the hypotheses are acceptable given the approximate beliefs. This probability could be determined experimentally for domains in which complete propagation is at least possible (even if it is not possible for real-time problem solving), and could be experimentally estimated for domains in which complete propagation is absolutely intractable. It may also be possible to directly derive this probability from appropriate domain and strategy models. This would be easier than trying to derive $P(acceptable(\hat{I}^G) \mid acceptable(\hat{I}^C) \wedge \hat{I}^G = \hat{I}^C)$ since it would not require modeling the relationship between the organization of agent interest areas and the uncertainty/incompleteness of the locally available data, and how such conditions interact with a particular control strategy.

For the corresponding conditional probability term in which $\hat{I}^G \neq \hat{I}^C$, there is the following bound:

$$P(acceptable(\hat{I}^G) \mid acceptable(\hat{I}^C) \wedge \hat{I}^G \neq \hat{I}^C) \geq$$
$$P(\forall I_j (I_j \neq \hat{I}^C) \Rightarrow (\forall h_i \in I_j, BEL(h_i) \geq AT) \mid acceptable(\hat{I}^C) \wedge I_j \neq \hat{I}^C)$$

Proof: The proof is analogous to the last proof.

$P(\forall I_j (I_j \neq \hat{I}^C) \Rightarrow (\forall h_i \in I_j, BEL(h_i) \geq AT) \mid acceptable(\hat{I}^C) \wedge I_j \neq \hat{I}^C)$ is the probability that all interpretations of the data other than the most likely interpretation have hypotheses whose true belief ratings would make them "acceptable" interpretations, given that the most likely interpretation has been found to be acceptable using a partial propagation strategy. While this does provide a lower bound on the component conditional probability, it is a weak bound because it is unlikely to be very large for reasonable values of the acceptance threshold AT. In the future we will be working to improve this bound by exploring how different strategies affect the likelihood of locating acceptable interpretations when such interpretations exist. This will also assist in characterizing the relative likelihood that the distributed system will find the same or different interpretations than the centralized system—a key issue in assessing the target conditional probability.

8 Conclusions

This paper began with a substantial discussion of SI and the kinds of interpretation and coordination strategies that can be used for FA/C distributed SI. Several basic results about the quality of solutions that can be produced by an FA/C-based distributed SI system were then presented. First, while it is possible for a distributed SI system to produce the same solution as a centralized system (and the true MPE of the available data) this requires strategies that are typically impossible

for real-world SI. Second, while coordination that focuses only on local solutions has been popular in FA/C systems and can be efficient, it is not guaranteed to produce centralized-equivalent or optimal solutions. Finally, if a satisficing approach to constructing solutions is adopted then an FA/C-based system can produce results that are "comparable" to those of a centralized system, but again only under conditions that are typically impractical.

Our future research will concentrate on pursuing the issues that appear in Sections 7.3 and 7.6. This is will involve characterizing the effects of practical approximate FA/C strategies for SI, based on probabilistic models of the domains. We believe that results based on the notion of approximate monotonicity will support some of the intuitions that people have had about approximate FA/C-based SI: certain types of hypotheses are better to construct and communicate, and it is useful to achieve a certain local level of belief in hypotheses before communicating them. In each of these cases we see that the likelihood of global solutions being correct when approximate strategies are used will be improved.

As we pointed out in the introduction, different coordination strategies for how and when to communicate the information necessary to resolve a GSOU can greatly affect the efficiency of the process. This issue is related to the question of local and global strategy interactions. For example, suppose that an agent wants to resolve a certain GSOU, but the appropriate external agent has not yet processed (interpreted) the data and created the hypotheses necessary to do this. Should the initiating agent wait for the external agent to get around to doing this? Should it instead be sent the raw data and do with it what it wants, even if that might produce a different result than would have been produced in the first scenario? Since our focus in this paper is on solution quality that can attained by FA/C distributed SI systems, we tried to separate the effects of the local and global strategies as much as possible. In practice, however, their interactions can have a profound effect on efficiency and solutions. These are other issues that we will be pursuing as part of our efforts to analyze the FA/C model.

References

[1] Yaakov Bar-Shalom and Thomas Fortmann, *Tracking and Data Association*, Academic Press, 1988.

[2] Norman Carver and Victor Lesser, "A New Framework for Sensor Interpretation: Planning to Resolve Sources of Uncertainty," *Proceedings of AAAI-91*, 724–731, 1991.

[3] Norman Carver, Zarko Cvetanovic, and Victor Lesser, "Sophisticated Cooperation in FA/C Distributed Problem Solving Systems," *Proceedings of AAAI-91*, 191–198, 1991.

[4] Norman Carver, Victor Lesser, "The DRESUN Testbed for Research in FA/C Distributed Situation Assessment: Extensions to the Model of External Evidence," *Proceedings of the International Conference on Multiagent Systems*, June, 1995.

[5] William Clancey, "Heuristic Classification," *Artificial Intelligence*, vol. 27, 289–350, 1985.

[6] Keith Decker, and Victor Lesser, "An Approach to Analyzing the Need for Meta-Level Communication," *Proceedings of IJCAI-93*, 360–366, 1993.

[7] Keith Decker, and Victor Lesser, "Designing a Family of Coordination Algorithms," *Proceedings of the International Conference on Multiagent Systems*, June, 1995.

[8] Victor Lesser and Daniel Corkill, "Functionally Accurate, Cooperative Distributed Systems," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 11, no. 1, 81–96, 1981.

[9] Victor Lesser and Daniel Corkill, "The Distributed Vehicle Monitoring Testbed: A Tool for Investigating Distributed Problem Solving Networks," *AI Magazine*, vol. 4, no. 3, 15–33, 1983 (also in *Blackboard Systems*, Robert Engelmore and Tony Morgan, editors, Addison-Wesley, 1988).

[10] Victor Lesser, "A Retrospective View of FA/C Distributed Problem Solving," *IEEE Transactions on Systems, Man, and Cybernetics*, special issue on Distributed Artificial Intelligence, vol. 21, no. 6, 1347–1362, 1991.

[11] Ann Nicholson and Michael Brady, "Dynamic Belief Networks for Discrete Monitoring," *IEEE Transactions on Systems, Man, and Cybernetics*, special issue on Knowledge-Based Construction of Probabilistic and Decision Models, vol. 24, no. 11, 1593–1610, 1994.

[12] Judea Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann, 1988.

[13] Yun Peng and James Reggia, *Abductive Inference Models for Diagnostic Problem-Solving*, Springer-Verlag, 1990.

APPENDIX E

IDP: A Formalism for Modeling and Analyzing Complex Interpretation Problem Domains *

Robert C. Whitehair and Victor R. Lesser
Department of Computer Science
University of Massachusetts
Amherst, MA 01003
413-545-3444(voice)
413-545-1249(fax)
Email: whitehair@cs.umass.edu

January 5, 1995

Abstract

This paper presents the *IDP formalism* for representing and analyzing complex interpretation problem domains. The formalism is being used to analyze the relationship between the performance of search-based interpretation problem solving systems and the inherent properties, or *structure*, of problem domains in which they are applied. Models built using this formalism can be used for describing, predicting and explaining the behavior of *interpretation* systems and for generalizing a specific problem solving architecture to other domains. Examples demonstrate how domain phenomena such as noise and missing data are represented, how non-local relationships between subproblems can be modeled, and how quantitative properties of search spaces can be calculated.

Keywords: control, search, mathematical foundations

*This material is based upon work supported by the National Science Foundation under Grant No. IRI-9321324 and the Office of Naval Research contract N00014-92-J-1450. The content does not necessarily reflect the position or the policy of the Government and no official endorsement should be inferred. This paper has not already been accepted by and is not currently under review for a journal or another conference. Nor will it be submitted for such during IJCAI's review preiod.

1 Introduction

In this paper, we present formal methods for characterizing the structure of *interpretation problem* domains[1] and their associated problem solving architectures. Our primary assumptions are that the performance of an interpretation problem solver applied in a specific domain is a function of the structure of the problem domain and that effective problem solving techniques can be derived from a formal specification of a problem domain. We define interpretation problems as a search process where, given an input string X, a problem solver attempts to find the most *credible* (or "best") explanation for X from the set I of all possible interpretations (i.e., an interpretation problem is a discrete optimization problem). Interpretation is a form of *constructive problem solving* based on *abductive inferencing*. Interpretation problems are similar to *classification* problems and to a more distant form of problem solving, *parsing*. In interpretation problems, the set I is constructed dynamically. This is in contrast to classification problems where I is preenumerated and the problem solving task involves only the identification of the best element of I [2].

For many domains, a distinct I can be specified in a naturally structured way and it is this structure that is exploited by control architectures to reduce the number of elements of I that must be generated or to otherwise increase the efficiency with which I is specified. Of particular interest to us are situations where I is finite and can be defined as the language generated by a grammar, G, and where the evaluation function, f, used during problem solving is recursively defined for interpretation trees i in I. In these situations, interpretations take the form of derivation trees of X and the constructive search operators used in interpretation problems are viewed as production rules of G. G, therefore, defines how interpretations are decomposed. For example, the production rule $p \rightarrow n_1 n_2 n_3$ might correspond to the interpretation "p is composed of an n_1, an n_2, and an n_3." Given an n_1, an n_2, or an n_3, a problem solver may invoke the search operator o_p to try and generate a p. Each syntactic rule of the grammar is associated with a corresponding semantic process which determines the actual "meaning" of an interpretation. This representation is similar to that used in the *Composite Decision Process (CDP)* model of Kanal and Kumar [6] and the semantic grammars used by Fu [4].

Based on this definition of interpretation, we define the *Interpretation Decision Problem (IDP)* formalism. The formalism comprises two components, one for modeling the domain phenomena that generate a specific problem instance and one for modeling the structure of an interpretation problem solver. The generational component of the IDP formalism will be referred to as IDP_G and the interpretation component will be referred to as IDP_I . The primary focus of this paper will be on IDP_G and the details of IDP_I are left to [7]. The IDP_G formalism models interpretation *domain theories*[2] in terms of four feature structures: *component* (or *syntax*),

[1] Intuitively, interpretation problems are tasks where a stream of input data is analyzed and an explanation is postulated as to what domain events occurred to generate the signal data – the problem solver is attempting to interpret the signal data and determine what caused it [3].

[2] A domain theory is the computational theory that is the basis for a problem solver's functionality.

utility (or *credibility*)[3], *probability* (or *distribution*), and *cost*. The different feature structures are represented in terms of a *domain grammar* and functions associated with production rules of the grammar. The formal definition of an IDP grammar follows.

Definition 1.1 *An IDP Grammar is a grammar,* $G_I =< V, N, SNT, S, P >$, where V is the set of terminal symbols, N is the set of nonterminal symbols, SNT is the set of *solution-nonterminal*[4] symbols that correspond to final states in the associated interpretation search space, S is the start symbol for the grammar, and P is the set of context-free production rules.

The component structure is modeled directly by the rules, P, of G. The component subproblems of A are defined by the right-hand-side (RHS) of a rule of the grammar, $p : A \rightarrow$(component subproblems). The full structure is specified recursively and subproblems specified by the nonterminals and terminals of a grammar correspond to states in the associated search space.

The credibility structure is defined recursively in terms of credibility functions associated with G's production rules. Credibility functions include consideration of the credibilities of the component elements and semantics. Given p. A $\rightarrow$ CD, the credibility of "A," f_{A}, is a function of the credibilities of "C" and "D" and the *semantic function*, Γ, that measures the degree to which "C" and "D" are semantically consistent. Semantics associated with each production rule determine the actual domain interpretation. The semantic functions will typically make use of grammar element attributes that are not used by syntactic functions. These attributes are represented using the *feature list convention* described by Gazdar, et al. [5]. For example, a semantic process in an acoustic vehicle tracking system might combine two partial vehicle tracks into a longer track using a complex process that verifies the consistency of the frequency characteristics of the partial tracks and the properties of the resulting track such as acceleration and velocity constraints based on time/location attributes of the tracks. Credibility functions are represented $f_{\mathrm{A}}(f_{\mathrm{C}}, f_{\mathrm{D}}, \Gamma_p(\mathrm{C}, \mathrm{D}))$, where the subscript of Γ, p, is the number of the corresponding production rule from the grammar. In a natural language interpretation system, a semantic function might determine the degree to which the combination of a noun phrase and a verb phrase is meaningful.

Similarly, computational cost structures are recursive functions of component elements and the cost of applying semantic operators. For example, given p. A $\rightarrow$ CD, the cost of generating "A," g_{A}, is a function of the cost of generating "C" and "D" and the cost of applying the semantic function $\Gamma_p(\mathrm{C}, \mathrm{D})$. This is represented $g_{\mathrm{A}}(g_{\mathrm{C}}, g_{\mathrm{D}}, C(\Gamma_p(\mathrm{C}, \mathrm{D})))$, where $C(\Gamma_p(i, j, ...))$ is the cost of applying the semantic function Γ_p.

[3]In [7], credibility structures are formally linked to the semantics associated with full and partial interpretations. Thus, a full or partial interpretation that has a high credibility can intuitively be thought of as having a highly consistent semantic interpretation and a full or partial interpretation that has a low credibility can intuitively be thought of as having an inconsistent or incomplete semantic interpretation.

[4]In the IDP formalism, a potential solution is referred to as an SNT. The uses and advantages of the SNT representation are discussed in [7].

The probability structure is defined in terms of the function, ψ. For each $p \in P$, $\psi(p)$ represents the distribution of RHSs associated with p. This can be used both to generate problem instances, in which case ψ models the likelihood of certain events happening in the environment, and to specify the actions available to a problem solver, in which case ψ models the problem solver's perception of and assumptions about the domain. For production p which decomposes to RHS_1, $\text{RHS}_2 \ldots \text{RHS}_m$, $\psi(p)$ models the frequency distribution with which these rules are used to generate problem instances. The frequency with which the rules are used in interpretation is a function of this distribution. Note that $\sum_n \psi(p.n) = 1$. ψ supports models of real-world phenomena such as uncertainty caused by noise, missing data, distortion and masking, as will be discussed in Section 2.

To our knowledge, this formalism is unique and it offers significant contributions to the study of interpretation problems. The most important is that the IDP formalism supports the analysis of the assumptions (implicit and explicit) that a problem solver makes about a domain. For example, using the IDP formalism, it is possible to analyze a problem solver's a priori expectations about a domain. By analyzing the statistical properties of the interpretation trees of a domain grammar, it is possible to determine characteristics that can be used during problem solving, such as a priori expected distributions of domain events that can be used in model-based processing. Furthermore, this analysis can also identify patterns and structures that can be used to construct effective problem solving strategies [7]. In addition, the IDP formalism supports comparative experiments and analyses involving different problem solvers applied in the same domain and the same problem solver applied to different domains. Consequently, this work represents an initial step needed to formalize complex search processes, such as those associated with blackboard systems [1] that are used in sophisticated interpretation systems.

The next section demonstrates how the IDP_G formalism can represent domain phenomena such as noise and missing data. Section 3 describes quantitative results that can be derived using the formalism and presents examples of how these can be used to define general domain structures.

2 Representing Problem Domains

The quality and effectiveness of analysis tools derived from a formal model of a problem domain are dependent on the degree to which the formalism adequately represents the phenomena, and their causes, characteristic of a domain. For interpretation domains, a modeling formalism must deal effectively with interacting phenomena, noise, missing data, distortion, and masking [3]. In this section, we will discuss how the IDP_G formalism addresses these issues.

The interpretation domain which we will study in detail is a vehicle tracking problem based on acoustic sensor data as defined in our previous work on RESUN [2]. The problem solver's input consists of preprocessed sensor data gathered from a single contiguous region. The problem solver then processes the data in an attempt to identify the type of vehicle that generated the signals and the path it traversed

through the region. For the vehicle tracking domain, a modeling formalism must also represent multiple events (i.e., multiple vehicle tracks) and patterned events (vehicles moving in a coordinated manner).

In order to model the genertion of a problem instance, the IDP_G simulator begins by creating a feature list for the start symbol, S. This list includes credibility, vehicle heading, velocity, type, etc. Next, the generator probabilistically chooses a production rule, p, associated with the start symbol and applies p to the start symbol. i.e., the generator randomly chooses a production rule with the start symbol on the left-hand-side and replaces S with the right-hand-side of the rule. The choice of which rule to apply is made based on the weights of the ψ associated with the different rules. As the generator replaces S, it assigns each of the replacing symbols a feature list. Each of the feature lists is determined by applying p's feature list functions to S's original feature list. The new symbols, each with their own, unique feature list, are added to either a queue, if the symbol is a nonterminal, or an output set, if the symbol is a terminal, and the whole process is repeated. The first element of the queue is removed and processed, and the resulting new symbols are added to the tail of the queue or to the output set. This continues until the queue is empty and the output set contains terminals with fully specified feature lists.

The basic structure of the vehicle tracking problem domain consists of *scenarios*. Each scenario consists of a number of *tracks*. Each track can represent a single vehicle traversing the region (an *I-Track*), a vehicle and a corresponding ghost track (a *G-Track*), or a group of vehicles moving in a coordinated manner (a *P-Track*). Each of these tracks is composed of *vehicle locations*, Vn, where n is a vehicle identifier. (e.g., V1 is a vehicle 1 location at time t.) Vehicle locations are composed of *group classes* and group classes consist of *signal classes*. The signal classes can be thought of as preprocessed signal data.

A grammar that will generate problem instances for this domain is shown in Fig. 1. These figures show the production rules of the grammar and the probability distribution values for ψ associated with each of the rules. The feature lists are shown as bracketed subscripts. In this example, "f" represents the characteristics time, x and y coordinates, velocity (v), acceleration (a), offset, and energy. Velocity is used to generate the x and y coordinates of the next location in the track. Acceleration is used to adjust velocity over time. Velocity and acceleration are also used to constrain the progression of a track. For example, each particular kind of vehicle track has a maximum velocity and acceleration that cannot be exceeded. Offset represents a displacement from a "base" position. This is used to generate pattern tracks of multiple vehicles and ghost tracks [7]. Figure 1 explicitly shows how feature lists are modified for the different grammar rules. For example, rule 10 indicates that the generation of two track locations from a pattern track location involves adding the offset to the x and y locations of track location T1. Similarly, the generation of a T1 track location from an I-Track1 in rule 4 also involves generating a "new" I-Track1 for the next time quantum (t+1) with new x and y positions (x+V+A, y+V+A). Many rules include a RHS of λ. This indicates that there is a possibilty that the production

1.	$S_{[f]}$	$\rightarrow$ $Tracks_{[f]}$	p=1
2.	$Tracks_{[f]}$	$\rightarrow$ $Tracks_{[f]}$ $Track_{[f]}$	p=0.1
		$\rightarrow$ $Track_{[f]}$	p=0.9
3.	$Track_{[f]}$	$\rightarrow$ I-Track1$_{[f]}$	p=0.25
		$\rightarrow$ I-Track2$_{[f]}$	p=0.25
		$\rightarrow$ P-Track1$_{[f]}$	p=0.10
		$\rightarrow$ P-Track2$_{[f]}$	p=0.10
		$\rightarrow$ G-Track1$_{[f]}$	p=0.15
		$\rightarrow$ G-Track2$_{[f]}$	p=0.15
4.	I-Track1$_{[f]}$	$\rightarrow$ I-Track1$_{[f,t+1,x+V+A,y+V+A]}$ $T1_{[f]}$	p=1
5.	I-Track2$_{[f]}$	$\rightarrow$ I-Track2$_{[f,t+1,x+V+A,y+V+A]}$ $T2_{[f]}$	p=1
6.	P-Track1$_{[f]}$	$\rightarrow$ P-Track1$_{[f,t+1,x+V+A,y+V+A]}$ P-T1$_{[f]}$	p=1
7.	P-Track2$_{[f]}$	$\rightarrow$ P-Track2$_{[f,t+1,x+V+A,y+V+A]}$ P-T2$_{[f]}$	p=1
8.	G-Track1$_{[f]}$	$\rightarrow$ G-Track1$_{[f,t+1,x+V+A,y+V+A]}$ G-T1$_{[f]}$	p=1
9.	G-Track2$_{[f]}$	$\rightarrow$ G-Track2$_{[f,t+1,x+V+A,y+V+A]}$ G-T2$_{[f]}$	p=1
10.	P-T1$_{[f]}$	$\rightarrow$ T1$_{[f,t,x+O,y+O]}$ T2$_{[f]}$	p=1
11.	P-T2$_{[f]}$	$\rightarrow$ T2$_{[f,t,x+O,y+O]}$ T2$_{[f]}$	p=1
12.	G-T1$_{[f]}$	$\rightarrow$ GT1$_{[f,t,x+O,y+O]}$ T1$_{[f]}$	p=1
13.	G-T2$_{[f]}$	$\rightarrow$ GT2$_{[f,t,x+O,y+O]}$ T2$_{[f]}$	p=1
14.	T1$_{[f]}$	$\rightarrow$ V1$_{[f]}$ N$_{[f]}$	p=1
15.	T2$_{[f]}$	$\rightarrow$ V2$_{[f]}$ N$_{[f]}$	p=1
16.	GT1$_{[f]}$	$\rightarrow$ GV1$_{[f]}$ N$_{[f]}$	p=1
17.	GT2$_{[f]}$	$\rightarrow$ GV2$_{[f]}$ N$_{[f]}$	p=1
18.	N$_{[f]}$	$\rightarrow$ n$_{[f]}$ N$_{[f]}$	p=0.1
		$\rightarrow$ n$_{[f]}$	p=0.25
		$\rightarrow$ λ	p=0.65
19.	V1$_{[f]}$	$\rightarrow$ G1$_{[f]}$ G3$_{[f]}$ G7$_{[f]}$	p=0.4
		$\rightarrow$ G1$_{[f]}$ G3$_{[f]}$	p=0.3
		$\rightarrow$ G1$_{[f]}$ G7$_{[f]}$	p=0.25
		$\rightarrow$ λ	p=0.05
20.	V2$_{[f]}$	$\rightarrow$ G3$_{[f]}$ G8$_{[f]}$ G12$_{[f]}$	p=0.4
		$\rightarrow$ G8$_{[f]}$ G12$_{[f]}$	p=0.3
		$\rightarrow$ G3$_{[f]}$ G12$_{[f]}$	p=0.25
		$\rightarrow$ λ	p=0.05
21.	GV1$_{[f]}$	$\rightarrow$ G-G1$_{[f]}$ G-G3$_{[f]}$ G-G7$_{[f]}$	p=0.2
		$\rightarrow$ G-G1$_{[f]}$ G-G3$_{[f]}$	p=0.3
		$\rightarrow$ G-G1$_{[f]}$ G-G7$_{[f]}$	p=0.25
		$\rightarrow$ λ	p=0.05
22.	GV2$_{[f]}$	$\rightarrow$ G-G3$_{[f]}$ G-G8$_{[f]}$ G-G12$_{[f]}$	p=0.4
		$\rightarrow$ G-G8$_{[f]}$ G-G12$_{[f]}$	p=0.3
		$\rightarrow$ G-G3$_{[f]}$ G-G12$_{[f]}$	p=0.25
		$\rightarrow$ λ	p=0.05
23.	G1$_{[f]}$	$\rightarrow$ S1$_{[f]}$ S2$_{[f]}$	p=0.45
		$\rightarrow$ S1$_{[f]}$ S3$_{[f]}$	p=0.1
		$\rightarrow$ S1$_{[f]}$ S4$_{[f]}$	p=0.1
		$\rightarrow$ S2$_{[f]}$ S3$_{[f]}$	p=0.1
		$\rightarrow$ S2$_{[f]}$ S3$_{[f]}$ S4$_{[f]}$	p=0.1
		$\rightarrow$ S1$_{[f]}$	p=0.05
		$\rightarrow$ S2$_{[f]}$	p=0.05
		$\rightarrow$ λ	p=0.05
24.	G3$_{[f]}$	$\rightarrow$ S5$_{[f]}$ S7$_{[f]}$	p=0.45
		$\rightarrow$ S5$_{[f]}$ S6$_{[f]}$	p=0.1
		$\rightarrow$ S6$_{[f]}$ S7$_{[f]}$	p=0.1
		$\rightarrow$ S4$_{[f]}$ S5$_{[f]}$	p=0.1
		$\rightarrow$ S7$_{[f]}$ S8$_{[f]}$	p=0.1
		$\rightarrow$ S5$_{[f]}$	p=0.05
		$\rightarrow$ S7$_{[f]}$	p=0.05
		$\rightarrow$ λ	p=0.05
25.	G7$_{[f]}$	$\rightarrow$ S11$_{[f]}$ S15$_{[f]}$	p=0.55
		$\rightarrow$ S11$_{[f]}$ S16$_{[f]}$	p=0.43
		$\rightarrow$ λ	p=0.02
26.	G8$_{[f]}$	$\rightarrow$ S13$_{[f]}$ S18$_{[f]}$	p=0.55
		$\rightarrow$ S13$_{[f]}$ S17$_{[f]}$	p=0.1
		$\rightarrow$ S14$_{[f]}$ S18$_{[f]}$	p=0.1
		$\rightarrow$ S15$_{[f]}$ S17$_{[f]}$	p=0.1
		$\rightarrow$ S13$_{[f]}$	p=0.05
		$\rightarrow$ S18$_{[f]}$	p=0.05
		$\rightarrow$ λ	p=0.05
27.	G12$_{[f]}$	$\rightarrow$ S6$_{[f]}$ S14$_{[f]}$ S17$_{[f]}$	p=0.45
		$\rightarrow$ S6$_{[f]}$ S14$_{[f]}$	p=0.25
		$\rightarrow$ S7$_{[f]}$ S14$_{[f]}$ S18$_{[f]}$	p=0.25
		$\rightarrow$ λ	p=0.05
28.	G-G1$_{[f]}$	$\rightarrow$ S1$_{[f]}$ S2$_{[f]}$	p=0.2
		$\rightarrow$ S1$_{[f]}$ S3$_{[f]}$	p=0.05
		$\rightarrow$ S1$_{[f]}$ S4$_{[f]}$	p=0.05
		$\rightarrow$ S2$_{[f]}$ S3$_{[f]}$	p=0.05
		$\rightarrow$ S2$_{[f]}$ S3$_{[f]}$ S4$_{[f]}$	p=0.05
		$\rightarrow$ S1$_{[f]}$	p=0.2
		$\rightarrow$ S2$_{[f]}$	p=0.2
		$\rightarrow$ λ	p=0.2
29.	G-G3$_{[f]}$	$\rightarrow$ S5$_{[f]}$ S7$_{[f]}$	p=0.2
		$\rightarrow$ S5$_{[f]}$ S6$_{[f]}$	p=0.05
		$\rightarrow$ S6$_{[f]}$ S7$_{[f]}$	p=0.05
		$\rightarrow$ S4$_{[f]}$ S5$_{[f]}$	p=0.05
		$\rightarrow$ S7$_{[f]}$ S8$_{[f]}$	p=0.05
		$\rightarrow$ S5$_{[f]}$	p=0.2
		$\rightarrow$ S7$_{[f]}$	p=0.15
		$\rightarrow$ λ	p=0.25
30.	G-G7$_{[f]}$	$\rightarrow$ S11$_{[f]}$ S15$_{[f]}$	p=0.30
		$\rightarrow$ S11$_{[f]}$ S16$_{[f]}$	p=0.30
		$\rightarrow$ λ	p=0.40
31.	G-G8$_{[f]}$	$\rightarrow$ S13$_{[f]}$ S18$_{[f]}$	p=0.15
		$\rightarrow$ S13$_{[f]}$ S17$_{[f]}$	p=0.05
		$\rightarrow$ S14$_{[f]}$ S18$_{[f]}$	p=0.05
		$\rightarrow$ S15$_{[f]}$ S17$_{[f]}$	p=0.05
		$\rightarrow$ S13$_{[f]}$	p=0.2
		$\rightarrow$ S18$_{[f]}$	p=0.25
		$\rightarrow$ λ	p=0.25
32.	G-G12$_{[f]}$	$\rightarrow$ S6$_{[f]}$ S14$_{[f]}$ S17$_{[f]}$	p=0.2
		$\rightarrow$ S6$_{[f]}$ S14$_{[f]}$	p=0.2
		$\rightarrow$ S7$_{[f]}$ S14$_{[f]}$ S18$_{[f]}$	p=0.25
		$\rightarrow$ λ	p=0.35
35.	n$_{[f,t]}$	$\rightarrow$ S1$_{[f]}$	p=0.05
		$\rightarrow$ S2$_{[f]}$	p=0.05
...	...	...	...
		$\rightarrow$ S20$_{[f]}$	p=0.05

Figure 1: Grammar Rules for a Vehicle Tracking Domain

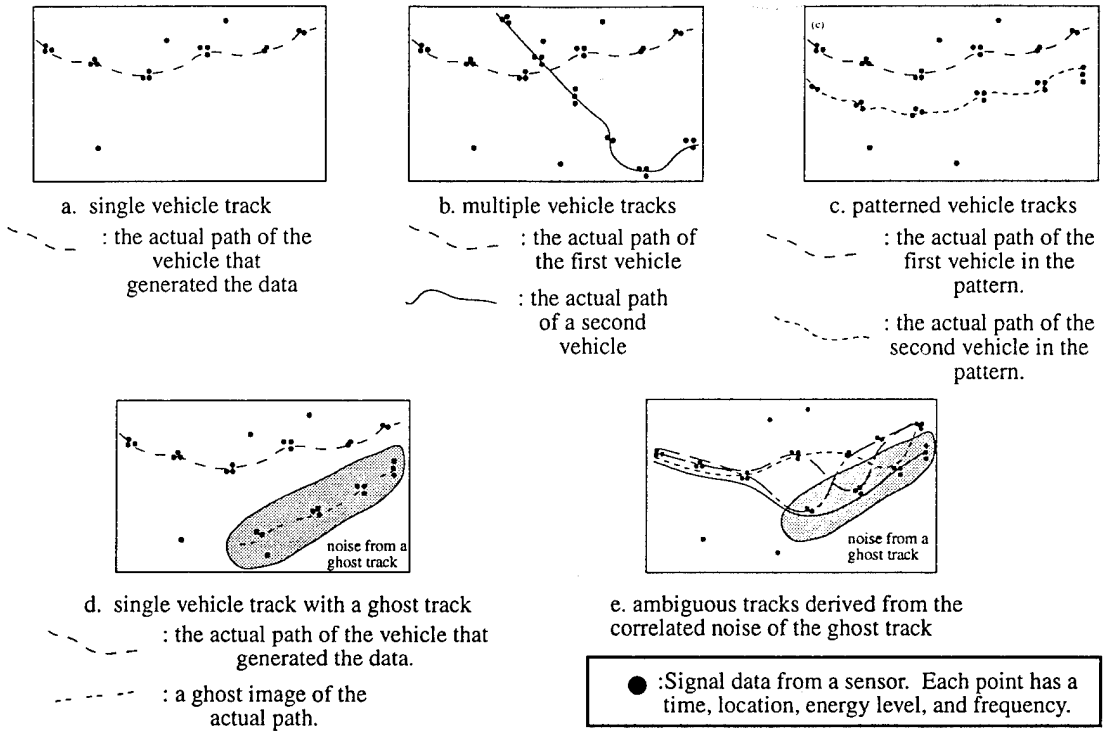


Figure 2: Vehicle Tracking Scenario Examples

rule will not lead to the production of signal data.

Rules 1 through 9 are used to generate the high-level track phenomena. The recursive rule number 2 will generate a number of track phenomena for each scenario. By adjusting the probabilities for the two alternative RHSs, this rule can be tailored to generate any number of tracks. If the probability of the first RHS is close to 1, this rule will generate many tracks. If it is close to 0, this rule will generate only one track. Rules 4 through 9 generate the time/vehicle locations for each track[5].

Simple scenario examples are shown in Fig. 2. In all these examples, the tracks shown move from left to right. This is a convention used for presentation clarity only. The actual grammar can generate tracks that originate anywhere on the sensed region's perimeter and at any point in time. Thus, if an experimental run is to simulate a time span of several days, the vehicle tracks that are generated can begin at any point in time. A single track is shown in Fig. 2.a. Figure 2.b is an example of multiple tracks in a single scenario. A pattern track is shown in Fig. 2.c. This is contrasted with the ghost track in Fig. 2.d. Notice that the ghost track differs from the pattern track in that the data is more "spotty." There is more missing data and more time instances for which data is altogether lacking.

2.1 Interacting Phenomena

The feature list convention is used to generate pattern and ghost tracks. This is shown in rules 6 through 9 and rules 10 through 13. These rules lead to domain events

[5]Note that these rules are not grounded. They continue to generate data until some predetermined time limit is reached or until the data generated falls outside of the region sensed by the problem solver.

consisting of two tracks with closely related properties. As shown in Figures 2.c and d, ghost tracks and pattern tracks move in coordination with each other. Ghost tracks do this because one signal is a reflection of a true signal and pattern tracks do this because they are composed of multiple vehicles moving according to a plan. For example, a pattern track might include a tanker vehicle and a second vehicle following it while refueling, multiple vehicles moving in an attack pattern, etc.

Rules 6 through 9 generate the data for each time-frame of the tracks. Rules 10 through 13 generate the patterned data. The vehicle patterns are generated using feature lists and offsets. Each element in a track, which will be referred to as a vehicle location, has an x and a y coordinate. As seen in rule 6, each element of a track has a basic x and y coordinate, but one element has a position modified by a variable *offset*. The offset can be determined by an arbitrarily complex function, resulting in a very expressive technique for generating coordinated domain events.

This is just one example of how the feature list convention can be used to generate domain phenomena that are coordinated or related in some way and that appear to interact through time and/or distance. This is a very important point because it suggests that, in some situations, it is possible to use a context-free grammar to generate problem instances that might be considered context-sensitive. The power of the analysis tools presented here is derived in large part from their context-free nature.

2.2 Noise

In any interpretation domain, noise plays a significant role in increasing the complexity of problem solving. This is the result of noise increasing the number of plausible interpretations (i.e., increasing the ambiguity) that have to be differentiated. We have developed techniques for modeling a variety of different types of noise and a formal definition.

Definition 2.1 *Noise* - A grammar, G, is subject to noise iff
$\exists$ a rule p $\in$ P, where p: A $\rightarrow$ uBv $\Rightarrow$ p$'$: A $\rightarrow n_1$un_2Bn_3vn_4, for u,v $\in$ (V $\cup$ N)*, A $\in$ N, B $\in$ (V $\cup$ N), $n_i \in$ (V $\cup$ N)*, and $f_p(f_u, f_B, f_v, \Gamma_p(u, B, v)) > f_{p'}(f_{n_1}, f_u, f_{n_2}, f_B, f_{n_3}, f_v, f_{n_4}, \Gamma_{p'}(f_{n_1}, u, f_{n_2}, B, f_{n_3}, v, f_{n_4}))$ for maximum ratings of the f_{n_i}. The distribution of p and p' is modeled by ψ.

Production rules 18 and 35, shown in Fig. 1, are used to generate *random noise*. This represents random phenomena associated with the natural state of the domain when no vehicles are passing through the region. There could be many causes of random noise including temperature changes, equipment malfunctions, natural flora or fauna, and more. The amount of random noise in a domain is determined by the probabilities associated with rule 18. If the probability of the first RHS is high, the domain will include a great deal of noise. If the probability is low, very little noise will be generated. The properties of the random noise can be further modified by adjusting the probabilities associated with the RHSs of rule 35.

In addition to adjustments to the probability distributions, random noise characteristics can be modified with the feature list convention. For example, note that the

feature lists used in rule 18 do not include any information about x and y locations. In our simulator, we determine the x and y locations of random noise using a uniform distribution function. This does not have to be the case. It would be easy to model a domain in which random noise tended to appear more in certain locations. Other properties of random noise, such as its energy level, can be manipulated in a similar way.

In addition to random noise, there is often noise that is more closely associated with the domain events that the system is trying to interpret. For example, in a vehicle tracking domain, it is reasonable to expect there to be phenomena corresponding to a vehicle interacting with the domain in an unexpected way. For example, a train may hit a rock or some other object on the track, something may fall off a ship, or a jet engine may emit some uncharacteristic noise. These sorts of phenomena are different from random noise, since they will only occur in the presence of a vehicle, but they are still a form of noise since they are not characteristic of a vehicle and may lead to ambiguity or otherwise obscure the sensing of phenomena that is characteristic of a vehicle. Rule 23 shows an example of how this sort of noise can be represented by adding elements to the RHS of an existing rule. Specifically, in rule 23,
$G1_{[f]} \rightarrow S2_{[f]}\ S3_{[f]}\ S4_{[f]}$, the extra signal, S4, could be considered noise.

More comprehensive noise structures can be represented at other levels of the grammar in a similar way. For example, ghosting phenomena is really a complex aggregate of noise at the track level. Noise phenomena can be easily added at the group level in a manner similar to that used to add signal data noise.

Another kind of noise is represented not by additional phenomena, but by slightly altered phenomena. This is sometimes referred to as sensor shifting. Intuitively, sensor shifting phenomena can be caused by errors in sensors or variations in a physical domain such as air or water temperature. All of the group level production rules include examples of sensor shifting. For example, the second RHS of rule 23 shows a shift from S2 to S3. Similar shifting can occur at all levels of the grammar.

2.3 Correlated and Uncorrelated Noise

In [7], we identified two kinds of noise, correlated and uncorrelated. Correlated noise leads to the generation of additional high-level interpretations. This ambiguity causes the problem solver to perform additional work to differentiate the possible interpretations. Uncorrelated noise may lead to additional work, but it does not lead to the generation of additional interpretations. Often, uncorrelated noise can be identified and eliminated from further processing with low-cost pruning operators [7].

The complexity of the example grammar is such that it is difficult to identify all correlated and uncorrelated noise by inspection alone. However, the rules associated with the non-terminal n include examples of both. For example, the first LHS of production rule 35: (see Fig. 1) $n_{[f,t]} \rightarrow S1_{[f]}$ is an example of correlated noise because the signal S1 can be used to build an interpretation of an *I-Track1* in situations where it would not otherwise be built. In contrast, the last LHS of production rule 35: $n_{[f,t]} \rightarrow S20_{[f]}$ is an example of uncorrelated noise because the signal data S20 is

not used to generate any vehicle tracks and can be ignored in subsequent processing.

A more meaningful example of correlated noise is shown in the problem instance illustrated in Fig. 2.e. In this example, production rule 18 from Fig. 1: $N_{[f,t]} \rightarrow n_{[f,t]}$ $N_{[f,t]}$ will create random noise throughout the sensed region. Should some of this noise occur near a track, it will cause the problem solver to create an interpretation that includes the noise as a possible element of the track. This interpretation could be in conflict with the interpretation that corresponds most closely with the event that created the noise.

2.4 Missing Data

Missing data is easy to represent and understand using IDP grammars. By simply dropping an element from an RHS, it is possible to model phenomena that results in data not being sensed. Missing data can be caused by sensor errors, environmental conditions, processing errors, and more. Formally,

Definition 2.2 *Missing Data* - A grammar, G, is subject to missing data iff $\exists$ p $\in$ P, where p: A $\rightarrow$ uBv $\Rightarrow$ p′: A $\rightarrow$ uv, for u,v $\in$ (V $\cup$ N)*, A $\in$ N, B $\in$ (V $\cup$ N),and $f_p(f_u, f_B, f_v, \Gamma_p(u, B, v)) > f_{p'}(f_u, f_v, \Gamma_{p'}(u, v))$. In addition, the distribution of p and p' is modeled by ψ.

The second and third RHSs to rule 19 show examples of how missing group level data can be modeled by omitting the appropriate non-terminal. The sixth and seventh RHSs of rule 23 show how missing signal data can be modeled in a similar way.

The production rules associated with ghost tracks also demonstrate how missing data phenomena can be related across production rules. Note that the RHSs for the ghost group level nonterminals are identical with the RHSs for the regular group level nonterminals. The only difference is in the distribution probabilities. For ghost data, the probability of missing signal data, or completely missing group data, is much higher. This is one of the ways in which ghost tracks can be identified. In this example, the related domain events, i.e., corresponding levels of missing data, are linked not by the feature list convention, but by a common non-terminal.

3 Characterizing Domain Structures

Given a formal IDP$_G$ model, it is possible to analytically characterize properties of the domain for use in explanation, prediction, and design of problem solver performance. In [7] we present analytical techniques for calculating a variety of different domain properties including expected cost of problem solving, expected frequencies of domain events, expected costs of individual interpretation search paths, the expected cost associated with incorrect search paths, ambiguity relationships, a quantified potential value for meta-operators, and more. Two specific domain structures are the concept of *marker* and *differentiator*. In the remainder of this section, we demonstrate the IDP$_G$ formalism by defining these two domain structures and we discuss how they can be used in dynamic control strategies and in the design of problem solver architectures.

The definitions of markers and differentiators are relative to the concept of a *solution nonterminal*, or *SNT*, described in Definition 1.1. For a given SNT, an intermediate result (i.e., a terminal or a nonterminal different from the SNT) is a *strong marker* if it is always (or, from a statistical perspective, almost always) *implied* by the SNT. Information about markers can be used to predict the occurance and nature of low-level events and, thus, can be used to support top-down, model-based processing. Furthermore, analysis of the component structure can be used to determine the degree to which an intermediate result *differentiates* an SNT. Differentiation refers to the extent to which an intermediate result is exclusively associated with an SNT. The differentiation relationship is the inverse of the marker relationship and it can be used to support bottom-up, island-driving processing.

To analytically determine marker and differentiator relationships for a given SNT, A, and a partial result, b, it is necessary to define several relationships between search state A and search state b. To accomplish this, it is necessary to think of a given IDP_G grammar as a definition of a search space in which the states belong to classes corresponding to terminals and nonterminals of the grammar and final states belong to the class of SNTs. For example, in the sample grammar, the nonterminal V2 defines a class of states in the search space. During problem solving, there could be many instances of the class, each associated with a different vehicle 2 location. Given the correspondance between elements of an IDP_G grammar and states in a search space, it is possible to compute relationships between states by computationally determining relationships between elements of the grammar.

The following definitions are needed to formally define the concepts of marker and differentiator.

Definition 3.1 $D(A) = P(S \vdash^* A)$, *where* $A \in \{SNT\}$. $D(A)$ defines domain specific frequency distribution functions for the set of SNTs, i.e., $D(A)$ = probability of the domain event corresponding to interpretation A occurring. In general, these distributions will be represented with production rules of the grammar associated with the start symbol. The RHSs of these rules will be from the grammar's set of SNTs (i.e., A $\in$ {SNT}). Uncertainty regarding this distribution leads to problem solving uncertainty.

For example, the SNTs for the vehicle tracking grammar are I-Track1, I-Track2, G-Track1, G-Track2, P-Track1, and P-Track2. From production rules 2 and 3 (see Fig. 1), the domain specific distribution function for the SNTs can be computed from the frequency of the nonterminal Track multiplied by the values for ψ corresponding to each of the SNTs. For example, $D(I-Track1) = 1.11*0.25 = 0.275$. Similarly, $D(I-Track2) = 0.275$, $D(G-Track1) = 0.11$, $D(G-Track2) = 0.11$, $D(P-Track1) = 0.165$, and $D(P-Track2) = 0.165$.

If the values for ψ in production rule 2 were to change, these values would be different. For example, if the ψ values for the RHSs of rule 2 were both 0.5, the distributions would be $D(I-Track1) = 2*0.25 = 0.5$, $D(I-Track2) = 0.5$, $D(G-Track1) = 0.2$, $D(G-Track2) = 0.2$, $D(P-Track1) = 0.3$, and $D(P-Track2) = 0.3$.

Definition 3.2 $\{RHS(A)\}$ = the set of elements that appear on right-hand-sides of production rules with A on the left-hand-side, $A \in N \cup SNT$.

From production rule 20, $\{RHS(V2)\} = \{G3, G8, G12\}$. A more complex rule is 26, $\{RHS(G8)\} = \{S13, S14, S15, S17, S18\}$.

Definition 3.3 $P(b \in \{RHS(A)\}^{\vdash+}) = \sum_{\forall i} P(b \in RHS_i(A)) + \sum_{\forall r'}(P(r' \in \{RHS(A)\}) * P(b \in \{RHS(r')\}^{\vdash+}))$, where $\{RHS(A)\}$ is the set of all RHSs of A, $RHS_i(A)$ is the i^{th} potential RHS of production rule of A, $P(b \in RHS_i(A)) = \psi(RHS_i(A))$ if $b \in RHS_i(A)$, 0 otherwise, each element r' is a nonterminal that appears in a RHS of A that does not also include b, $b \in V \cup N \cup SNT$, and $A \in N \cup SNT$. The probability of partial interpretation b being included in any RHS of A, as defined by the distribution function $\psi(A)$. The "$\vdash^+$" notation indicates that the definition of RHS is recursive. i.e., $\{RHS(A)\}^{\vdash+}$ represents the transitive closure of all states that can be generated from A. Thus, b can be in an RHS of A, or in the RHS of some element of an RHS of A, etc.

From production rules 20, 24, 26, and 27,

$$\{RHS(V2)^{\vdash+}\} = \{G3, G8, G12, S4, S5, S6, S7, S8, S13, S14, S15, S17, S18\} \quad (1)$$

$$P(S4 \in \{RHS(V2)\}^{\vdash+}) = \quad (2)$$

$$P(G3 \in \{RHS(V2)\}) * P(S4 \in \{RHS(G3)\}^{\vdash+}) = 0.65 * 0.1 = 0.065 \quad (3)$$

Definition 3.4 $P(A \vdash^* b) = D(A) * P(b \in \{RHS(A)\}^*), A \in SNT, b \in V \cup N$. Probability that the partial interpretation, b, is generated from full or partial interpretation A, where b is a descendant of A.

Definition 3.5 *Ambiguity* – Given a domain event, A, its interpretation is *ambiguous with* the interpretation of a second domain event, B, when B *subsumes* A (the subsume relationship is specified in [7]). i.e., A is ambiguous with B when the low-level signal data generated by B can be mistaken for an A. Note that this definition of ambiguity is *not* reflexive. Thus, A being ambiguous with B does not imply that B is ambiguous with A.

Definition 3.6 $P(A \cap b) = P(A \vdash^* b) + \sum_{\forall B} P(B \vdash^* b), A, B \in SNT, b \in V \cup N$, and where the interpretation of A is *ambiguous with* the interpretation of each B. Intersection of domain events A and b, where b is a descendant of A. The intersection of A and b will occur when both A and b are generated during the course of a specific problem solving instance. This will occur when A leads to the generation of b and when the occurrence of a distinct event, B, leads to the generation of b and when A is ambiguous with B. In the case where B leads to the generation of b, b and A still intersect because an A will be generated during processing since A is ambiguous with B.

Definition 3.7 $P(a \cap b) = P(a) * P(b \in \{RHS(a)\}^{\vdash+}), a \in N, b \in V \cup N$. Intersection of domain events a and b, where b is a descendant of a and where a is a nonterminal that is not an SNT.

Definition 3.8 $P(b) = \sum_{\forall A} D(A) * P(b \cap A \mid A), A \in SNT, b \in V \cup N \cup SNT.$ The probability of partial interpretation b being generated. The notation $P(b \cap A \mid A$ is the probability that b and A intersect in situations where event A is responsible for the generation of the signal data.

Given these definitions, it is now possible to formally define the concept of marker:

Definition 3.9 $MARKER(A, b) = \frac{P(A \cap b)}{D(A)}$, where A is an element of the set SNT and b is any terminal or nonterminal. The value of the MARKER function indicates the probability that the generation of an A will cause the generation of a b. A value of MARKER(A, b) that is close to 1 indicates a relationship where every occurance of A indicates that a b can be derived. The value of MARKER(A, b) indicates the strength of the relationship between A and b.

Definition 3.10 $MARKER(a, b) = \frac{P(a \cap b)}{P(a)}$, where a is a nonterminal and b is any terminal or nonterminal.

Knowledge of markers can be used to design a problem solving architecture and to construct focusing mechanisms in a dynamic control strategy. For example, assume that the vehicle tracking system is attempting to extend a track of type I-Track1 through a large amount of noise spread over a wide region. This could be computationally expensive if the problem solver has to examine the implications of every possible point of noise in a bottom-up fashion. The number of potential tracks, and the cost, will increase combinatorially with the amount of noise. Information about markers can be used to design control strategies to reduce search costs in these situations. In the sample grammar, the strongest marker at the group level for an individual component of an I-Track1 (i.e., a vehicle location, V1) is G1. Specifically,

$$MARKER(V1, G1) = P(V1 \cap G1)/P(V1) = \tag{4}$$

$$(P(V1) * P(G1 \in \{RHS(V1)\}^{\vdash +}))/P(V1) = P(G1 \in \{RHS(V1)\}^{\vdash +}) = 0.95. \tag{5}$$

This information could be used to design an expectation driven control strategy for extending an I-Track1 (from t_i to t_j) by predicting the characteristics of all G1's that can be used to extend the track to time t_{j+1} and then only following the implications of the G1's that match the predictions. Also, this information could be used in differential diagnosis processing to disambiguate competing hypotheses [2]. For example, in certain situations it is possible to use marker information to determine if a hypothesis was erroneously derived from noise. By examining a hypothesis' supporting data, a problem solver can determine if the support from a strong marker is consistent with expectations. If the support is either much less or much greater than expected, this would be a good indication that the hypothesis is not correct.

For example, assume that, for hypothesis X, $MARKER(X, y) = 0.99$. If the problem solver generates an X for which there is no y supporting data, the problem solver can conclude that the probability of this happening is $1 - 0.99$ and that the X might be erroneous. In the vehicle tracking domain, this might occur if there is a signal group, s, that is a strong marker for a particular kind of track, t. If the problem solver can derive an interpretation of a t without finding any s signals, it is likely that the hypothesis is incorrect.

In contrast, the signal data S15 is a poor marker for the group level event G8.

$$MARKER(G8, S15) = P(G8 \cap S15)/P(G8) = \quad (6)$$

$$(P(G8) * P(S15 \in \{RHS(G8)\}^{\vdash +}))/P(G8) = P(S15 \in \{RHS(G8)\}^{\vdash +}) = 0.1. \quad (7)$$

A formal definition of the differentiator relationship can be given as:

Definition 3.11 $DIFF(A, b) = \frac{P(A \cap b)}{P(b)}$ where $A \in$ SNT, and $b \in V \cup N \cup SNT$ is any terminal or nonterminal. A value of DIFF(A, b) that is close to 1 indicates a strong causal relationship between A and b to the exclusion of all other causes. A value of DIFF(A, b) that is close to 0 indicates a weak causal relationship between A and b.

Knowledge about differentiators can be used both in the design of problem solving architectures and dynamic control algorithms. Architecturally, differentiators can be used to construct special operators for differential diagnosis [2]. In control algorithms, differentiators can be used to focus problem solving activity [3].

In the vehicle tracking grammar, S1, S2, and S3 are strong differentiators for the event V1 (a vehicle location of type 1). Specifically,

$$DIFF(V1, S1) = P(V1 \cap S1)/P(S1) = \quad (8)$$

$$(P(V1) * P(S1 \in \{RHS(V1)\}^{\vdash +}))/P(S1) = 0.55 * 0.70/0.39 = 0.99. \quad (9)$$

(Note that S1 is also generated by production rule 35, and, consequently, $P(S1)$is greater than the probability of S1 being derinved solely from V1.) Similarly, DIFF($V1, S2$) $= 0.55 * 0.70/0.39 = 0.99$ and DIFF($V1, S3$) $= 0.55 * 0.3/0.17 =$ 0.97. Given a large amount of noise, a possible control strategy would be to determine if there was a large amount of S1, S2, and S3 events in the data. If there are, there is a strong likelihood that they were generated by a track with V1 as a component. This information could be used to filter out data that are weak markers for V1. For example, MARKER(V1,S8) = 0.07. Consequently, ignoring S8 data might be a reasonable strategy in this situation.

In general, the marker and differentiator relationships can be used to explain the success of techniques such as approximate processing and incremental planning [1]. From a bottom-up perspective, the differentiator relationship can identify the intermediate results that are most appropriate for abstracting and clustering. From a top-down perspective, the marker relationship can help predict the expected characteristics of intermediate results derived from model-driven processing and thus focus processing by filtering out noise.

Though not discussed in this paper, the concepts of marker and differentiator can be further refined to include information about the credibilities of the hypotheses. For example, the relationships could vary significantly for low-credibility events and high-credibility events. Given a track level hypothesis that has a low-credibility, its relationship with markers could be quite different from that of a hypothesis with similar characteristics and a high-credibility. The same is true for differentiator relationships. A low-level hypothesis with a high-credibility may be a much better differentiator than a similar hypothesis with a low-credibility.

4 Conclusion

The techniques presented in this paper for formally modeling interpretation domains represent a necessary step toward developing a basis for explaining, predicting, and designing the performance of sophisticated knowledge-based problem solvers. By explicitly modeling a problem domain, the IDP_G formalism can be used to experimentally test the effects of alternative control strategies. For example, a specific strategy can be applied in different domains by using slightly altered versions of an IDP_G grammar. This could be done to identify the characteristics of domains in which the control strategy is most effective. Alternatively, different control strategies can be compared and contrasted statistically by using an IDP_G grammar to generate numerous experimental problem instances.

More importantly, by explicitly specifying a problem domain, IDP_G formalism makes it possible to analyze its characteristics in ways that can be used to design control strategies. For example, general domain structures such as differentiators and markers can be identified and used to design meta-operators. These structures represent expectations that are derived from statistical abstractions of a domain.

References

[1] Norman Carver and Victor Lesser. The Evolution of Blackboard Control. *Expert Systems with Applications*, 7(1), 1991. Special issue on The Blackboard Paradigm and Its Applications.

[2] Norman Carver and Victor R. Lesser. Planning for the control of an interpretation system. *IEEE Transactions on Systems, Man, and Cybernetics*, 23(6), 1993. Special Issue on Scheduling, Planning, and Control.

[3] Lee D. Erman, Frederick Hayes-Roth, Victor R. Lesser, and D. Raj Reddy. The Hearsay-II speech-understanding system: Integrating knowledge to resolve uncertainty. *Computing Surveys*, 12(2):213–253, June 1980.

[4] King Sun Fu. *Syntactic Pattern Recognition and Applications*. PH, 1982.

[5] Gerald Gazdar, Geoffrey K. Pullum, and Ivan A. Sag. Auxiliaries and related phenomena in a restrictive theory of grammar. *Language*, 58(3):591–638, 1982.

[6] Vipin Kumar and Laveen N. Kanal. The CDP: A unifying formulation for heuristic search, dynamic programming, and branch-and-bound. In L. Kanal and V. Kumar, editors, *Search in Artificial Intelligence*, Symbolic computation, chapter 1, pages 1–27. Springer-Verlag, 1988.

[7] Robert C. Whitehair and Victor R. Lesser. A Framework for the Analysis of Sophisticated Control in Interpretation Systems. Technical Report 93–53, Department of Computer and Information Science, University of Massachusetts, Amherst, Massachusetts 01003, 1993.

APPENDIX F

Learning Coordination Plans in Distributed Problem-Solving Environments

Toshiharu Sugawara
NTT Basic Research Labs.
3-1 Wakamiya, Morinosato
Atsugi, Kanagawa 243-01
Japan
sugawara@ntt-20.ntt.jp

Victor Lesser*
Department of Computer Science
University of Massachusetts
Amherst, MA 01003
USA
lesser@cs.umass.edu

Abstract

Coordination is an essential technique in cooperative, distributed multi-agent systems. However, sophisticated coordination strategies are not always cost-effective in all problem-solving situations. This paper presents a learning method to acquire coordination plans for specific problem-solving situations so that the appropriate type of coordination strategy is used. This learning is accomplished by recording and analyzing traces of inferences after problem solving. The analysis results in identification of situations where inappropriate coordination strategies have caused redundant activities or the lack of timely execution of important activities, thus degrading system performance. Based on this identification, situation-specific coordination plans are created which use additional non-local information about activities in the networks to remedy the problem. An example from a real distributed problem-solving application involving diagnosis of a local area network is described.

CFP Topics: Learning and Adaptation in Multi-agent Worlds; Coordination; Organizational Self-Design

*This research was supported in part by a grant from ARPA contract N00014-92-J-1698, an ONR contract N00014-92-J-1450, and a grant from Network General Corporation. The content of the information does not necessarily reflect the position or the policy of the organizations supporting this research, and no official endorsement should be inferred.

1 Introduction

Achieving globally coherent activity in a cooperative, distributed multi-agent system is a difficult problem for a number of reasons. One difficulty is that an agent's control decisions, based only on its local view of problem-solving task structures, may lead to inappropriate decisions about which activity it should do next, what results it should transmit to other agents and what results it should ask other agents to produce. If an agent has a view of the task structures of other agents it can make more informed choices [6, 7, 11]. Another difficulty is that even with the availability of this type of meta-level information, there is still residual uncertainty about outcomes of tasks and what future tasks will be coming into the system which may result in agents still exhibiting non-coherent behavior. These difficulties with achieving effective coordination are further exacerbated by the fact that an agent, in acquiring and exploiting a non-local view of other agents' activities, may expend significant computational resources. This expense is in terms of communication delays waiting for this information to arrive, as well as the computational cost of both providing this information in a suitable form to other agents and processing this information to make local control decisions. Thus, for specific problem-solving situations, due to the inherent uncertainty in agents' activities and the cost of meta-level processing, it may not be worthwhile to acquire a complete view of other agents' activities, and thus some level of non-coherent activity may be the optimal coordination strategy [11].

For example, coordination to avoid redundant activities may be unnecessary if processing resources are not overloaded and if the communication channel is neither expensive nor overloaded. In this case, local problem solving is done more efficiently where there is no additional overhead for coordination. If a coordination strategy can be developed whose costs can be varied depending upon the amount and type of non-local information it uses to make coordination decisions, then it seems that only a selected, possibly situation-specific, view of other agents' activities is necessary. The obvious next question is how to determine what the appropriate situation-specific view is and what type of coordination rules should be used in the situation. It is our hypothesis that for many multi-agent applications, especially those operating in complex, open and possibly evolving environments, it is very difficult or impossible for the designer of a system to *a priori* anticipate all the problem-solving contexts and exactly which information and what coordination strategy for each context will be most cost-effective. Thus, in this paper, we propose integrating into each agent a distributed learning component that agents can use to acquire through experience which information is

most effective for a specific situation and how to exploit this information.

In the remainder of the paper, we develop this distributed learning component in detail, based on EBL techniques [5, 12] using a limited domain model and inductive techniques such as comparative analysis [9], and discuss the implementation of these ideas in a real distributed problem-solving system, LODES [15, 16], which performs internetwork diagnosis.

2 An Example Problem

The LODES network diagnosis system observes message traffic on the network in order to detect and analyze situations which indicate a hardware or software problem in the network. The LODES system becomes a multi-agent system when the network is implemented as multiple communication segments. In this case, a separate LODES agent is placed on each segment and is responsible for monitoring traffic on that segment of the network and diagnosing any problems that are recognized. LODES agents start out with the assumption that there is no need to explicitly coordinate with other LODES agents on different segments of the network (i.e., their subjective views used for coordination are based only on their local task structure). The only interaction among agents is an occasional request for a diagnostic task that can be performed by only the LODES agent that is located on a specific network segment and the sharing of information about the network configuration and the final results of diagnosis. The basic assumption behind this lack of coordination is that there are sufficient communication and computational resources available on the network to sustain a certain level of non-coherent behavior (e.g., two agents diagnosing the same problem). This assumption is appropriate for diagnostic activities in most network environments and was the basis for how the system was originally implemented. However, it is not a valid assumption, for example, in network environments where diagnostic activities generate a lot of message traffic, and where there are multiple agents performing redundant diagnosis and the cost of communication is significant. Consider the following situation in the network environment as shown in Fig. 1, where L1 to L7 are LODES agents, Net1 to Net7 are network segments, and Net5 and Net6 are connected with a narrow-bandwidth line. Suppose a host, H_A on Net1, sends a broadcast to all hosts on Net7, but most of the hosts cannot understand that protocol. Upon receiving the broadcast, they simultaneously send back error packets in order to inform H_A that they have discarded its broadcast packet (this problem is called the first problem). In this situation, multiple agents L1 ... L7 will be concurrently diagnosing the same problem since all will see the messages indicating the discarding of the broadcast packets. In responding to

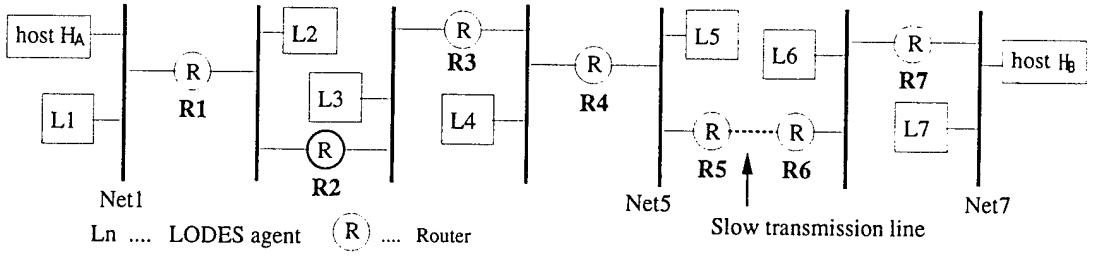


Figure 1: Network Environment for the Example Problem

this problematic situation, each LODES agent may independently send diagnostic test packets through the network to understand its cause, which could be for a variety of reasons. This redundant diagnostic message traffic may in turn lead to another problematic situation if the network environment has network segments which are implemented as narrow-bandwidth lines, or if a tariff is associated with each message, or if there are other problems occuring in the network that also need to be diagnosed and computational resources are being used redundantly. This is a problematic situation since it indicates the potential overloading of an expensive or scarce resource. This secondary problem, which is directly caused by LODES agents' activities, can be attributed to the lack of effective coordination among agents.

Note that one of the interesting aspects of our diagnosis application is that, in this example problem, the LODES agents, in their monitoring of the network, will detect the secondary problem. In this case, agents L5 and L6 will decide that the secondary problem is not tolerable because they know the existence of the narrow-bandwidth line. The existence of this problem and the fact that it was caused by the LODES agents themselves will be the trigger for the learning component to be invoked.

3 The Learning Framework

Our approach to learning coordination rules can be described in the coordination model proposed by Decker and Lesser [3, 4] based on generic coordination relationships. In this model, each agent makes scheduling decisions based on a subjective view of its own and other agents' task structures and the relationships among these tasks (such as *enable*-, *facilitate*-, and *support*-relations[1]) and

[1]This is a new relationship that was not in the original formulation discussed in [3] and relates to how a task in one agent can effect the subjective view of another agent's task structure by changing the importance rating of its tasks. This rating change can, in turn, cause the agent to choose one task over another for execution.

resource usage patterns (such as *use*-relation). These task and resource relationships are not only qualitative but also have quantitative characteristics. In certain situations this subjective view will lead to ineffective/inappropriate actions because certain non-local interrelationships have not been appropriately taken into account; this can occur because the limited subjective agent view does not include an important relationship or the quantitative character of the relationship is incorrectly inferred because of the use of a default assumption or out-of-date/incorrect information.

Before presenting our learning framework, we introduce the following terminology. During problem solving, an agent selects a *goal* and, to achieve this goal, a *task* is executed. A task consists of a partially ordered set of *subtasks*. Each subtask also has an associated *subgoal*. The execution of a task means that all the subtasks are executed in a manner consistent with the specified partial order. A subtask may further be divided into smaller subtasks. The individual task that cannot be divided into subtasks is called an *operation* (in [3], this is called a *method*), which must be an executable form. There are usually a number of different tasks that can be used to achieve a goal. The LODES planner specifies the alternative task structures to achieve a specific diagnostic goal and then selects an appropriate task among them. This selected task is called a *plan*. The local scheduler selects an operation that should be executed next based on the plan. During this execution, communication activities such as transmitting an operation's outcome or requesting information may occur.

The knowledge used by our learning framework includes a heuristic collection of rules and procedures for 1) recognizing situations where there is costly incoherent behavior[2], 2) identifying control decisions that lead to this behavior, and 3) modifying these control decisions or replacing them with new decision processes that rectify the inappropriate control. When an undesirable situation is observed, the learning component first identifies the *mainstream tasks and messages*, which contributed to achieving the final result of the LODES diagnostic process. It then locates in the trace which tasks induced the observed undesirable situation. Next, agents rebuild the situation model they had locally in the diagnostic process. These models are exchanged and agents generate a more comprehensive model about the observed situation. Agents then reproduce the inference process based on this model. Two cases are possible depending on the result of the

[2]This is done by the meta-level controller within an agent or by an external monitor for specific (shared) resources (such as network resource or database). What kinds of events are detected are described in [17]. It is also assumed that an agent locally records an abstracted trace of its recent problem-solving actions which can be reviewed on-line by the learning component.

reproduction. The first case, which we will call a *lack-of-information* problem, can be solved in a very local way by just choosing an alternative task to satisfy the given diagnostic goals. In this case, control rules are added to the planner to choose the alternative task for satisfying the given goal in this particular situation. In essence, the learning component determines what aspects of the task and resource structure, which if taken into account in the scheduling of local activities would lead to more coordinated behavior; it then creates the rule to acquire or calculate this data (i.e., the agent's subjective view of the relationship between its activities and those of other agents is extended). The second case, which we will call an *incorrect-control* problem, is not solved solely by making existing rules more context sensitive with respect to the external agent environment. In this case, control rules for a new situation-specific scheduling strategy are added to increase the ratings of appropriate tasks or alter the coordination strategy used by agents in this specific case.

This basic paradigm of monitoring, analysis and replanning (learning, in this case) is very similar to the one laid out but never implemented for use in organizational self-design [2] and a partially implemented approach to meta-level control for a single-agent system [8].

4 Learning Steps

This section details the steps of the learning process using the example problem discussed previously.

4.1 Recording Inference Traces

An agent must be able to record traces of reasoning for analysis by the learning component. Traces have to contain data describing the following aspects of local problem solving: (1) executed tasks and operations; (2) task relationships; (3) information communicated; (4) resource usage (if possible); (5) domain data used; (6) domain knowledge used; and (7) control knowledge used. All this information is stored with a time stamp. An agent must be able to reproduce the same decisions and reasoning from the recorded trace.

4.2 Mainstream Tasks and Messages

The first step in the learning process is to identify the mainstream tasks and messages. These can be obtained by tracing backwards through task relations from the final result in the problem-solving trace as follows. The task that directly led to the final result is a mainstream task. A task that has a task relation with a mainstream task is a mainstream task. A message that produced a mainstream

```
Condition:
   execution-order(T1 T2 T3) ;; Order of executions
   mst(T1),mst(T3)           ;; T1 and T3 were the Mainstream Tasks.
   not-mst-between(T1 T3)    ;; No mainstream tasks were executed after T1 and before T3.
   task-proposed-by(T3,M1)   ;; T3 was proposed based on message M1.
   task-proposed-by(T2,T1)   ;; T2 was proposed based on the result of T1.
   arrive-when(M1,T2)        ;; M1 arrived before T2 was executed.
   msg-status(M1) = Analysis-Postponed ;; The rating of the message was so low
                                       ;; that its analysis was postponed.
Then: message-has-the-low-rating(M1)
```

Figure 2: An Example of Learning Heuristics
This rule describes the situation where the execution of a mainstream task (T3) is delayed (task T2 that does not contributed to the final result is executed before T3) because an important message (M1) has been lowly rated.

task is a mainstream message. A task that produced a mainstream message is a mainstream task. A task that produced the content of a mainstream message is a mainstream task. This process of computing the mainstream extends beyond a single agent's local problem solving to incorporate activities of agents distributed throughout the network.

4.3 LAP Detection

The second step of the learning process is to isolate situations in which the following inappropriate actions occurred.

(D1) Execution of unnecessary actions; tasks that do not belong to the mainstream set are redundant or unnecessary for achieving the goal of problem solving for the current episode.

(D2) Tasks that were not executed in a timely manner; these are usually caused by inappropriate task allocation, delay of communications, or by executions of unnecessary tasks.

(D3) Longer than expected task durations; this situation is usually caused by resource overloads or by inappropriate scheduling in other agents.

(D4) Redundant mainstream task; for example, a variable is defined twice or its value is redundantly sent from other agents.

(D5) Tasks that induced problematic external actions; see the example problem.

These problem-solving situations are called *learning analysis problems* (LAPs) and indicate situations where non-coherent behavior has taken place.

For example, the rule in Fig. 2 shows the situation where the mainstream task `T3` has not been executed in a timely manner because the message `M1`, which would cause the task `T3` to be instantiated and executed, has been inappropriately rated causing the processing of message `M1` to be delayed.

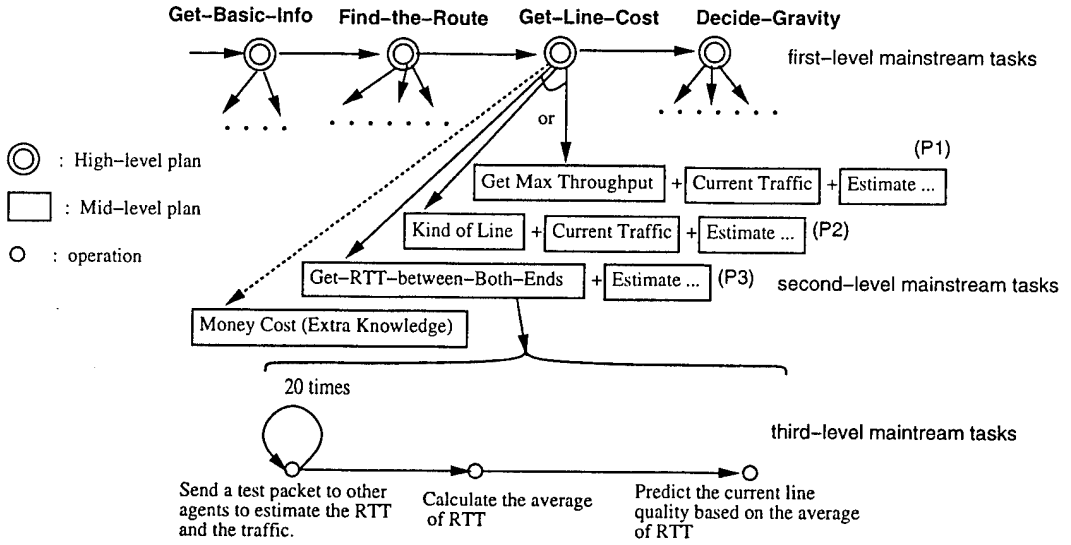


Figure 3: Plans created during diagnosis of the example problem

In the example problem, the LAP detection identifies the task(s) in which the test packets were sent by analyzing the trace of the first diagnostic problem solving (e.g., an example of a D5 situation). Fig. 3 illustrates the mainstream tasks. Agents can identify that the observed test packets were sent in the plan P3 named "Get-RTT-between-Both-Ends" in this step.

4.4 Task Structure Analysis

The third step of the learning process involves agents building a more comprehensive and objective model to describe the detected LAP.

(A1) Each agent creates the (subjective) model about the task structure and resource usages when the LAP occurred.

(A2) The agent identifies the reason why the task causing the LAP was selected. The model created in A1 can be thought of as subjective because it is based on what information an agent used in its deliberations—not what information was actually available.

(A3) All involved agents exchange their subjective model and generate more comprehensive models. These models include goals, scheduled tasks, inference states and resource usages of all agents as well as other observed domain data. These models are furthermore extended by adding results of non-mainstream tasks. These results are also exchanged.

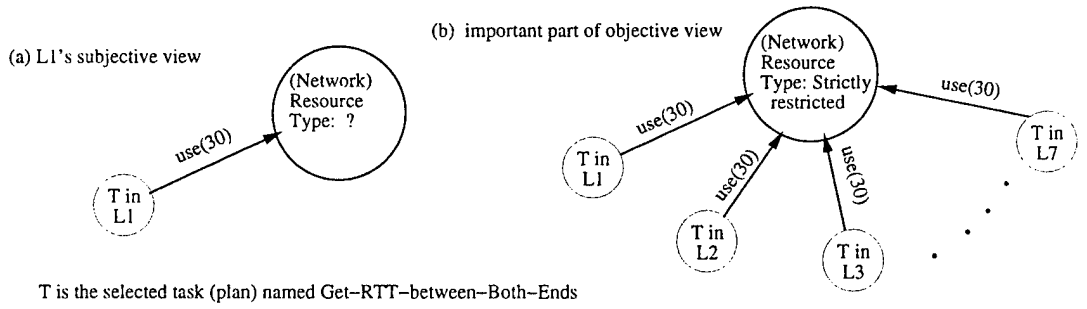
(A4) The agent that executed the task causing the LAP reproduces the inference using the generated comprehensive model. Depending on whether or not the agent selects the same task, the LAP can be seen as being caused by either the lack-of-information or the incorrect-control problem as described in Section 3. If it is the lack-of-information problem, the agent puts marks on pieces of the comprehensive model that are used to select the appropriate task.

In the example problem, agents create a comprehensive model about the situation in which the plan "Get-RTT-between-Both-Ends" was selected. Agents select this plan to estimate the network load and bandwidth by gathering statistics on the round-trip time (RTT) of a number of test packets sent into the network. This plan, which is not the optimal way to gather this statistic, produces only an approximate value for RTT. A better plan would have been to use the Simple Network Management Protocol (SNMP), which is a communication protocol designed for acquiring network management data [1], but this was not possible in the example since the adjacent network routers did not implement it. An agent, in order to understand what other tasks agents are executing, must know that local routers do not use the SNMP, and that all agents along the route from Net 7 to Net 1 will be performing the same diagnoses. Agents can then understand that the identical task was selected in other agents. This view of which tasks were to be executed in other agents was not present and thus responsible for the LAP occurring; this is one of the important parts of the enhanced subjective view necessary for achieving more coordinated activities (see Fig. 4 (a) and (b)).

4.5 Coordination Control Modification

The fourth step involves adding or modifying control rules for coordination based on the analysis of the model developed in the previous step. The approach to modification depends on the types of problems identified in the previous step. A lack-of-information problem occurs when an agent has the control rule to select another appropriate task but (1) lacks sufficient time to acquire and analyze necessary information or (2) lacks rules for acquiring necessary information. In the former case, EBL techniques can be used to develop a more efficient version of the control rule that meets the timing constraints. In the latter case, rules for acquiring the missing information are added. These types of rules may include requesting/sending data and changing the order of communication/task executions.

For an incorrect-control problem, agents have to generate a new rule or modify an existing rule in order to reach the alternative control using the following



T is the selected task (plan) named Get–RTT–between–Both–Ends

(c) The learned rule

IF Min–of–Max–Thrput < 64Kb ;; that is, the type of resource is strictly restricted
and the plan "Get–RTT–between–Both–Ends"
is selected in more than two agents ;; that is, the model of the current situation is identical to (b)
Then
Share the result of the plan "Get–RTT–between–Both–Ends."

Figure 4: Models for Problem Solving: (a) is the model created in L1 before learning. This model does not include the resource type and other agent's intended tasks. (b) is the model that will be created in L1 ... L7 after learning, and (c) is the rule finally derived by learning. The model (b) includes important data involved in coordination activities. This model is created to apply the rule (c); To check its "if-part," first the value of "Min-of-Max-Thrput" is calculated then other agents' planned tasks are gathered. Then the learned coordination control takes place when the acquired model is identical to the model (b).

learning knowledge:

(M1) **Changing the rating of the specific goal and message** so that the appropriate task is selected in the specific situation. For example, the LAP found by the rule in Fig. 2 will be fixed by increasing the rating of the message M1 by its sender or receiver.

(M2) **Changing the order of operations and communications** since some operations are often locally commutative. The purpose of this change is to speed up the execution of another agent which is waiting for the result of an operation that can be done earlier in the local agent without any consequence.

(M3) **Allocating a task to another idle agent** in order to take advantage of concurrent execution.

(M4) **Using results calculated by another agent** in order to avoid redundant actions. In this situation, one agent executes the corresponding task and others wait for the result.

(M5) **Selecting another (sub)task** that is expected to produce the same outcome but does not contain the operation that caused the LAP.

In the example problem, if agents have the control rule that can derive appropriate (at least, different) coordinated actions from the comprehensive model, only the rule to acquire information about the resource type and usages by other agents is generated (lack-of-information problem). It is possible, however, that agents cannot select another task based on the comprehensive model. This is because the model does not include the resource type information and/or agents do not have the control rules to select another task (incorrect-control problem). In such a case, agents modify their control by applying (M4) so that the LODES agents performing the same diagnostic activity can share the results of the plan executed by a single agent (see Fig. 4).

4.6 Situation Identification

Since the learned rules are situation-specific, agents have to identify the situation in which the learned rule should be applied. The purpose of this step is to create the part of the rule for identifying the situation. The situation must be identified based on the comprehensive model constructed in step four. However, a complete version of this comprehensive model is usually over-specific; we think that only a part of this model is essential to characterize the situation where the derived rule should be applied. The question we then face is what part of the comprehensive model is necessary to uniquely identify the problematic situation.

An inductive method is introduced to solve this identification problem. First, the past traces and other agents' traces containing the same LAP as well as the last reasoning trace are gathered[3], then these traces are divided into two types of instances—positive and negative instances. For example, if a LAP is identified by the rule in Fig. 2, the traces include the situation where T1, T2, T3 and M1 are bound to the same values are gathered. If the same LAPs are developed, the traces are positive; if no or different LAP are developed, they are negative. When a variable used in the different types of instances has the same value or when a variable used in the same type of instances has different values, we can conclude that the variable is not necessary for distinguishing the situation from others. This process of comparison is called *comparative analysis* (CA) [8].

In this example, however, since L1 ... L7 diagnose the same secondary problem in this example, agents can identify the situation by comparing their traces

[3]This approach assumes that we have stored some amount of history in the agent about past problem-solving experiences in a way to facilitate such analysis [17].

Table 1 (a): Comparative Analysis (L5 and L6)

Variables	values in L5	values in L6	
Adjacent networks	Net4, Net6	Net5 and Net7	eliminated
Adjacent routers	R4, R5	R6, R7	eliminated
End-nodes	(L1, L7)	(L1, L7)	
Src-MAC	xx:xx:xx:f:2a:3b	xx:xx:xx:0:12:8c	eliminated
Dst-MAC	xx:xx:xx:f:2a:3a	xx:xx:xx:0:2f:7e	eliminated
Type-of-Storm	ICMP-Echoes	ICMP-Echoes	
MaxThrput1	10,000,000	64,000	eliminated
MaxThrput2	64,000	10,000,000	eliminated
Min-of-MaxThrput (Quantitative Measure)	64,000	64,000	64,000 (max of L5 and L6's values)
Observed-Number-of-Echoes (Quantitative Measure)	68	61	61 (min of L5 and L6's values)
Current-Traffic	low	low	

Table 1 (b): Comparative Analysis (L4 and L5)

Variables	values in L4	values from the previous CA	
End-nodes	(L1, L7)	(L1, L7)	eliminated
Type-of-Storm	ICMP-Echoes	ICMP-Echoes	eliminated
Min-of-MaxThrput (Quantitative Measure)	10,000,000	64,000	
Observed-Number-of-Echoes (Quantitative Measure)	68	61	eliminated (almost identical)
Current-Traffic	low	low	eliminated

We assume that L5 claimed the secondary problem. L6 also claimed the same problem thus its trace is the positive instance. L4 did not so its trace is negative. All variables are used in the diagnosis of the secondary problem. Only the variable Min-of-MaxThrput that indicates the minimum of the local throughput to adjacent network segments remains.

of reasonings about the secondary problem. Although L1 to L7 observe the secondary problem, only L5 and L6 concluded that the problem was not tolerable, because they knew the existence of a narrow-bandwidth line. This difference is expressed by the variable Min-of-MaxThrput; other variables are eliminated by the CA (cf. Table 1). The results finally obtained through learning are illustrated in Fig. 4.

Note that after this learning, an agent's action may still cause the same or

another problem. For example, the situation where the learned rules should be applied cannot correctly be identified because of insufficient instances. Thus the LAP will occur again. In another case, the appropriate inference strategy will change over time in a gradually evolving environment. In both cases, however, by applying this learning iteratively, agents can incrementally converge on the correct rules.

5 Empirical Results

All aspects of the learning component—except the EBL (we assume that a conventional EBL package can be imported in the future)—have been implemented in the LODES system. The learned control plan for the example problem described in this paper represents the output of our implemented learning component. The CPU time and the elapsed time of the learning process for this example are approximately 5.12 seconds and 102.5 seconds in C and LISP (interpreter) on Sun machines (SparcStation 1 (SS1), SS1+ and SS2)[4]. The difference between the CPU time and the elapsed time is caused by communications and synchronization. The modified agent control resulted in a decrease of approximately 10% in the time for all agents to arrive at a diagnosis of the first problem because of additional coordination activities. The total number of communication packets sent among agents is almost identical to the amount before the learning, but those that passed through the slow line are approximately half. We want to emphasize that quality inference is more important than the efficiency in this example problem. Before the learning is applied, agents almost blindly create their plans based on the subjective view illustrated in Fig. 4 (a), which did not include the view of non-local agents; thus, the environment caused the secondary problem. The learning enables the agents to create their plans based on more accurate and appropriate views about agent networks, task relations and their environments as illustrated in Fig. 4 (b). An efficiency-oriented example is described in [17].

6 Conclusion

Although coordination is an essential technique for cooperative distributed problem solving, there needs to be a balance between too much or too little coordination since both can degrade overall system performance. The problem is how to create for a specific situation the appropriate model of the network state, and the necessary processing of this information to arrive at an acceptable control decision. We feel it is an impossible task for the system designers, at design time,

[4]The CPU time can be improved by compiling and/or optimizing the LISP program.

to choose this appropriate balance in all situations, especially for systems operating in open and evolving environments. This paper has discussed a learning method for acquiring situation-specific coordination control rules. The method enables the system to avoid a previously recognized non-coherent situation by modifying/extending local control strategies to be more situation-specific so as to have a more enhanced view of the task relations and the environment. This approach to learning in a multiagent system is quite different in character from reinforcement learning approaches or statistical approaches to learning rules and parameters done by other researchers [10, 13, 14].

References

[1] Case, J., Fedor, M., Schoffstall, M. and Davin, J. "A Simple Network Management Protocol (SNMP)," RFC1157, 1990.

[2] Corkill, D. D. and Lesser, V. R., "The Use of Meta-Level Control for Coordination in a Distributed Problem Solving Network" (long paper), *Proc. of the Eighth International Joint Conference on Artificial Intelligence*, August 1983, pp. 748-756, Karlsruhe, FRG. (Also published in *Computer Architectures for Artificial Intelligence Applications*, Benjamin W. Wah and G.-J. Li (eds.), IEEE Computer Society Press, pp. 507-515, 1986.)

[3] Decker, K. and Lesser, V. R. "Quantitative Modeling of Complex Environments," *International Journal of Intelligent Systems in Accounting, Finance and Management*, special issue on Mathematical and Computational Models of Organizations: Models and Characteristics of Agent Behavior, 1993.

[4] Decker, K.S. and Lesser, V.R. "Designing a Family of Coordination Algorithms," in *Proceedings of the 13th International Workshop of Distributed Aritificial Intelligence*, July 1994.

[5] DeJong, G. "Generalizations Based on Explanations," *Proc. of 7th IJCAI*, pp. 67-69, 1981.

[6] Durfee, E. H., Lesser, V. R. and Corkill, D. D., "Coherent Cooperation Among Communicating Problem Solvers," *IEEE Transactions on Computers*, Vol. 36, Issue 11, November 1987, pp. 1275-1291. (Also published in *Readings in Distributed Artificial Intelligence*, A. Bond and L. Gasser (eds.), Morgan Kaufmann Publishers, California, 1988, pp. 268-284.)

[7] Durfee, E. H. and Lesser, V. R., "Partial Global Planning: A Coordination Framework for Distributed Hypothesis Formation," *IEEE Transactions on Systems, Man, and Cybernetics*, 21(5):1167-1183, September/October 1991.

[8] Hudlická, E. and Lesser, V. R., "Meta-Level Control Through Fault Detection and Diagnosis," *Proc. of the 1984 National Conference on AI*, 1984, pp. 153-161.

[9] Hudlická, E. and Lesser, V. R., "Modeling and Diagnosing Problem-Solving System Behavior," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 17, Number 3, May/June 1987, pp. 407-419. (Also published in Readings in Distributed Artificial Intelligence, A. Bond and L. Gasser (eds.), Morgan Kaufmann Publishers, California, 1988, pp. 490-502.)

[10] Kinney, M. and Tsatsoulis, C.,"Learning Communication Strategies in Distributed Agent Environments," Working Paper CECASE-WP-93-4, Center for Excellence in Computer-Aided Systems Engineering, The University of Kansas, Lawrence, 1993.

[11] Lesser, V. R., "A Retrospective View of FA/C Distributed Problem Solving," *IEEE Transactions on Systems, Man, and Cybernetics*, 21(6):1347-1362, Nov/Dec. 1991.

[12] Mitchell, T. M., Keller, R. M. and Kedar-Cabelli, S. T. "Explanation-Based Generalizations: A Unifying View," *Machine Learning*, Vol. 1, pp. 47-80, 1986.

[13] Sen, S., Sekaran, M. and Hale, J., "Learning to coordinate without sharing information," *Proc. of AAAI94*, pp. 426-431, 1994.

[14] Shoham, Y. and Tennenholtz, M., "Emergent Conventions in Multi-Agent Systems: initial experimental results and observations," *Proc. of KR-92*, 1992.

[15] Sugawara, T. "A Cooperative LAN Diagnostic and Observation Expert System," *Proc. of IEEE Phoenix Conf. on Comp. and Comm.*, pp. 667-674, 1990.

[16] Sugawara, T. and Murakami, K. "A Multiagent Diagnostic System for Internetwork Problems," *Proc. of INET'92*, Kobe, Japan, 1992.

[17] Sugawara, T. and Lesser, V. R. "On-Line Learning of Coordination Plans," COINS Technical Report, 93-27, Univ. of Massachusetts, 1993. (The shorter version of this paper is also published in *Proc. of the 12th Int. Workshop on Distributed AI*, 1993.)

To: Regional Director
Team Leader
ACO

This technical report was sent to me by DTIC because it does not include the DD-1498 form with the proper disclosure/distribution statement.

Please obtain this form with proper instructions and return it and the technical report directly to DTIC.

Also implement procedures with the contractor to correct this problem.

Thank You,

Jim Carbonara,
Director, Field Operations